User Guide
for the

APM-08

Keithley MetraByte Corporation

* * * * *

The
APM-08
AD/DA Converter
for Apple II Plus & IIe

Revision A, - 1985

Part Number: 24850

KEITHLEY METRABYTE CORPORATION

440 MYLES STANDISH BLVD., Taunton, MA 02780
TEL. 508/880-3000, FAX 508/880-0179

Warranty Information

All products manufactured by Keithley MetraByte are warranted against defective materials and worksmanship for a period of one year from the date of delivery to the original purchaser. Any product that is found to be defective within the warranty period will, at the option of Keithley MetraByte, be repaired or replaced. This warranty does not apply to products damaged by improper use.

Warning

Keithley MetraByte assumes no liability for damages consequent to the use of this product. This product is not designed with components of a level of reliability suitable for use in life support or critical applications.

Disclaimer

Information furnished by Keithley MetraByte is believed to be accurate and reliable. However, the Keithley MetraByte Corporation assumes no responsibility for the use of such information nor for any infringements of patents or other rights of third parties that may result from its use. No license is granted by implication or otherwise under any patent rights of Keithley MetraByte Corporation.

Notes

Keithley MetraByte/Asyst/DAC is also referred to here-in as *Keithley MetraByte*.

Basic™ is a trademark of Dartmouth College.

IBM® is a registered trademark of International Business Machines Corporation.

PC, **XT**, **AT**, **PS/2**, and **Micro Channel Architecture**® (MCA) are trademarks of International Business Machines Corporation.

Microsoft® is a registered trademark of Microsoft Corporation.

Turbo C® is a registered trademark of Borland International.

Table of Contents

Chapter 1

INTRODUCTION

1.1 SUMMARY OF APM-08 FUNCTIONS.

MetraByte's APM-08 is an 8 channel 12 bit high speed A/D converter, timer/counter, digital I/O, and 2 channel 12 bit D/A board for the Apple II Plus and IIe computers. The APM-08 board is a full length board and can be plugged into any of the Apple expansion slots. All connections are made through a 40 pin header and flat cable assembly through the rear of the computer. An optional screw connector board (STA-AP) facilitates making connections outside the computer. The following functions are implemented on the APM-08:-

1. An 8 channel, 12 bit successive approximation A/D converter with sample/hold. The full scale input of each channel is +/-5 volts with a resolution of 0.00244 volts (2.44 millivolts). Inputs are single ended with a common ground and can withstand a continuous overload of +/-30 volts and brief transients of several hundred volts. All inputs are fail safe i.e. open circuit when the computer power is off. A/D conversion time is typically 25 microseconds (35 microseconds max.) and depending on the speed of the software driver, throughputs of up to 30,000 channels/sec are attainable.

2. 2 channels of 12 bit multiplying D/A converters are provided. These may be used with an on-board fixed -10v precision reference voltage for an output of 0 - +10v or with an external positive or negative reference voltage up to +/-10v or with an A.C. reference e.g. 400Hz for synchro/resolvers. The output is the product of the input reference and the digital input. The digital input data is double buffered for single step update. Output settling time is typically 30 microseconds to 0.01% for a full scale step.

3. An 8254 programmable counter timer provides periodic interrupts for the A/D converter and can additionally be used for event counting, pulse and waveform generation, frequency and period measurement etc. There are three separate 16 bit down counters in the 8254. One of these (Counter 2) is connected to the system clock, and all I/O functions of the remaining two are accessible to the user. Input frequencies up to 8MHz can be handled by the 8254.

4. 7 bits of TTL digital I/O are provided composed of one output port of 4 bits and one input port of 3 bits.

5. 1 precision -10.00v (+/-0.1v) reference voltage output is derived from the A/D converter reference. This output can source/sink 2mA.

6. An external interrupt input is provided to allow user programmed interrupt service routines to provide background data acquisition or interrupt driven control. The APM-08 includes status and control registers that make interrupt handshaking a simple procedure. The interrupt input may be externally connected to the timer/counter or any other trigger source.

7. Apple buss power (+5, +12 & -12v) is provided along with all other I/O connections on the rear connector. This makes for simple addition of user designed interfaces, input signal conditioning circuits, expansion multiplexers etc.

The APM-08 is easily programmed as a memory mapped peripheral using assembly language or PEEKS and POKES in BASIC. An on-board 2K EPROM (2732) provides a high level Applesoft interface e.g. PR#2:PRINT"C4" will perform a conversion on Channel 4 (see Chapter 4). The ROM provides access to all board functions with the exception of interrupt driven processes which are not supported by Applesoft BASIC. Using state of the art data conversion components, the APM-08 has been designed to provide a powerful and inexpensive analog/digital interface on a single board. It is ideally suited to any application requiring high speed 12 bit data acquisition at low cost. The freedom from complexity and the individual memory locations of each function make programming straightforward. Applications include data logging, process control, signal analysis, robotics, energy management, product testing, digitizers and touch screens, laboratory and medical instrumentation etc. A system block diagram appears in Fig. 1.1.

An optional screw terminal board (MetraByte STA-AP) housed in a plastic instrument case can be mounted outside the computer and greatly simplifies connection of the APM-08 to your application. All I/O lines from the APM-08 are connected to miniature screw terminal connectors. The digital I/O port lines are monitored by L.E.D.'s and a small breadboard area with +/-12v & +5v power is available for amplifiers, filters, and other user supplied circuits.

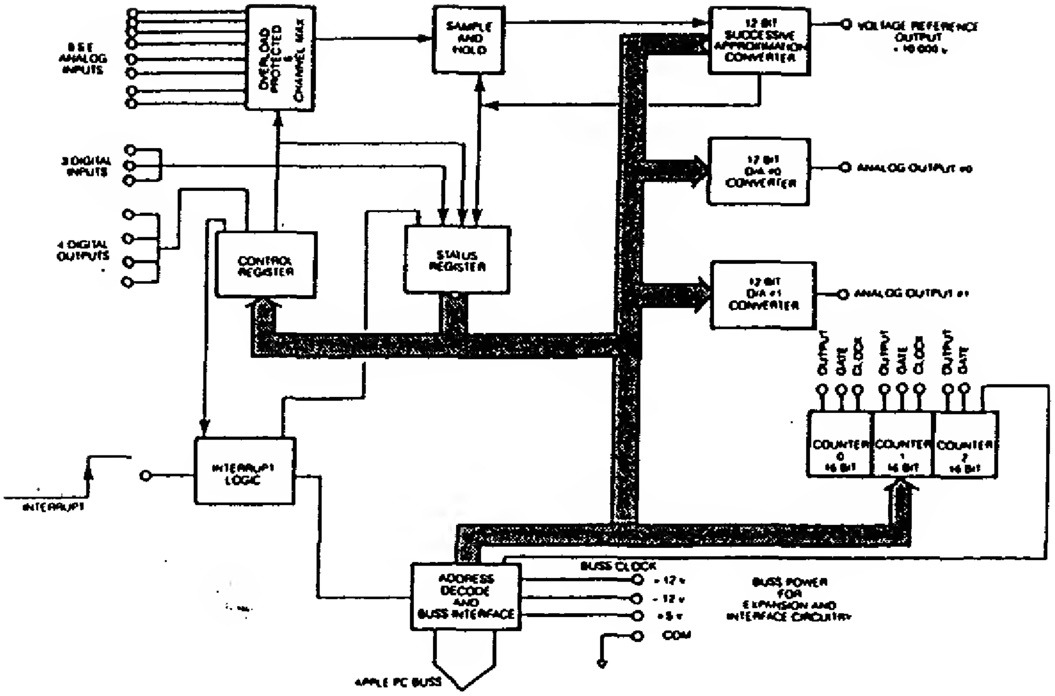


Fig. 1.1 Block Diagram of APM-08

Chapter 2

INSTALLATION

2.1 HARDWARE INSTALLATION

The APM-08 may be plugged into any of the expansion slots inside your Apple II Plus or IIe **except** slot 0 which is reserved for system use. There are no switches or jumpers on the board that need setting before installation.

Turn off the power on your computer, and remove the top cover of your computer. Before you touch or handle any of the computer electronics or the APM-08 board make sure you have discharged any static charge that your body may have acquired. The easiest way to do this is to momentarily touch the case of the Apple power supply or backplate on the computer (assuming it is grounded). Next, remove the xAPM-08 from its protective electrostatic packaging and place it in a vacant slot (except 0). If you are connecting the flat cable, plug the header into the rear of the board and lead the cable out through one of the larger slots of the rear panel. Both ends of the cable are polarized and interchangeable. If the strain relief on the plug interferes with an adjacent board then snap it off, it is not essential but simply improves the mechanical integrity of the cable.

When you have finished installation, replace the top cover of your computer and check that it boots up normally.

Remember, TURN OFF THE POWER whenever installing or removing any peripheral board including the APM-08. Failing to observe this precaution can cause costly damage to the electronics of your computer and/or the APM-08 board.

If for any reason you later remove the APM-08 board, MetraByte recommends that you retain the special electrostatically shielded packaging and use it for storage.

Chapter 3

PROGRAMMING

3.1 PROGRAMMING APM-08

At the lowest level, APM-08 is programmed using memory I/O instructions. In BASIC these are the POKE X,Y and PEEK(X) functions. Assembly language and most other high level languages have equivalent instructions. Use of these functions usually involves pre-formatting data and dealing with absolute memory addresses. Although not demanding, this can require many lines of code and necessitates an understanding of the devices, data format and architecture of the APM-08. To simplify programming for many applications an on-board ROM driver is supplied that operates with Applesoft BASIC using simple commands. These commands are described in Section 3.8. The tradeoff involved in using the on-board ROM is a loss in speed due to the execution time of interpreted BASIC and an inability to use interrupt driven functions (background data acquisition) as interrupts are not supported by BASIC. The compensating simplicity of programming may in many cases be more important than obtaining the ultimate in performance through assembly language routines. The hardware supports both approaches.

3.2 MEMORY ADDRESS MAP OF APM-08

First of all let's take a look at the memory address map of the APM-08:-

ADDRESS	READ	WRITE
Slot I/O base + 0	A/D Lo byte	Start 8 bit A/D conversion
+ 1	A/D Hi byte	Start 12 bit A/D conversion
+ 2	APM-08 status	APM-08 control register
+ 3	-	-
+ 4	Read Counter 0	Load Counter 0

+ 5	Read Counter 1	Load Counter 1
+ 6	Read Counter 2	Load Counter 2
+ 7	-	Counter control reg.
+ 8	-	D/A #0 Low byte
+ 9	-	D/A #0 High byte + Load
+ 10	-	D/A #1 Low byte
+ 11	-	D/A #1 High byte + Load

The various device addresses use the peripheral card I/O space (see Apple Technical Reference Manual). The reserved locations for peripheral I/O are as follows:-

SLOT NUMBER	SLOT I/O BASE ADDRESS	
	(Hex)	(Decimal integer)
0	$C080	-16256
1	$C090	-16240
2	$C0A0	-16224
3	$C0B0	-16208
4	$C0C0	-16192
5	$C0D0	-16176
6	$C0E0	-16160
7	$C0F0	-16144

The following example shows how to read the status register of a board in slot # 4:-

```
xxx10 X% = PEEK(-16190)  :REM -16190 = Base + 2
```

To perform similar operations for the A/D, D/A's and timer-counter, we need to know more about the format of the data for these devices. This is discussed in the following sections.

3.3 STARTING THE A/D CONVERTER

An A/D conversion is initiated by writing to location SLOT BASE ADDRESS + 0 or SLOT BASE ADDRESS + 1. To simplify further explanations the variable BASE will be used as the value of the SLOT BASE ADDRESS. If you write to BASE + 1, a full 12 bit A/D conversion is performed. Writing to BASE initiates a short cycle 8 bit conversion. A 12 bit conversion takes no more than 35 microseconds to complete, a short cycle 8 bit conversion takes less time and will not exceed 25 microseconds. (These times are dependent on the type and manufacturer of AD574 A/D converter used in your APM-08 and may be less, but will not exceed the durations specified).

Starting an A/D conversion:-

12 bits	xxx10 POKE BASE + 1, 0
8 bits	xxx10 POKE BASE, 0

The value of the data written to these locations is irrelevant and is lost. It is only needed to satisfy the syntax of BASIC. The decoded address write pulse is in fact what starts the A/D.

3.4 READING THE A/D DATA

After the end of conversion the data from the A/D may be read from locations BASE and BASE + 1. Data follows a low byte/high byte sequence which corresponds to the way the 6502 handles 16 bit word data. The data is left justified, so that BASE + 1 contains the most significant 8 bits from the conversion:-

BIT POSITION	D7	D6	D5	D4	D3	D2	D1	D0
(BASE + 1)	B1 (MSB)	B2	B3	B4	B5	B6	B7	B8

The remaining 4 least significant bits followed by 4 zeroes are read from BASE:-

BIT POSITION	D7	D6	D5	D4	D3	D2	D1	D0
(BASE + 0)	B9	B10	B11	B12 (LSB)	0	0	0	0

The left justification allows you to read 8 or 12 bit data to 8 bits of resolution by simply accessing one byte.

The A/D data bits B1-B12 correspond to an offset binary code:-

BINARY	HEX	ANALOG INPUT VOLTAGE
0000 0000 0000	000	-5.0000 v (-Full scale)
0000 0000 0001	001	-4.9976 v
. . .	.	.
. . .	.	.

0100 0000 0000	400	-2.5000 v (-1/2 scale)
. . .	.	.
. . .	.	.
1000 0000 0000	800	+/-0 v (zero)
1000 0000 0001	801	+0.0024 v
. . .	.	.
. . .	.	.
1100 0000 0000	C00	+2.5000 v (+1/2 scale)
. . .	.	.
. . .	.	.
1111 1111 1111	FFF	+4.9976 v (+Full scale)

A sequence of BASIC PEEK() instructions to read the data would be:-

```
xxx10  XL% = PEEK(BASE)        :REM  read low byte
xxx20  XH% = PEEK(BASE + 1)    :REM  read high byte
xxx30  X% = XH%*16 + XL%/16    :REM  combine bytes, X% = data
```

Note the use of integer variables throughout. BASE can be integer as well as all the data which is always in the range 0 - 4095. From this point you can turn the data in bits into volts or other engineering units (start using real variables!):-

```
xxx40  V = X%*10/4096          :REM  output * span/resolution
xxx50  V = V - 5               :REM  subtract zero offset, -5.0000 v
```

If we were using an input amplifier or attenuator with gain G, we could add another line to provide scaling etc.:-

```
xxx60  V = V * G
```

3.5 THE APM-08 STATUS REGISTER

The status register provides information on the operation of APM-08. It is a read only register at I/O location BASE + 2 and has the following format:-

BIT POSITION	D7	D6	D5	D4	D3	D2	D1	D0
(BASE + 2)	EOC	IP3	IP2	IP1	IRQ	MA2	MA1	MA0

The bits have the following significance:-

EOC: End of Conversion. If EOC is high (Logic 1) the A/D is busy performing a conversion. Data should not be read in this condition as it will be invalid. Wait for the EOC to return to logic 0 signifying valid data available.

IP3 - IP1: These bits correspond to the three digital input port lines IP3,IP2 and IP1. They may be used for any digital data input.

IRQ: After generation of an interrupt to the processor IRQ is set to logic high (1). It is reset to logic low (0) by a write to the control register. This provides a means of acknowledging or "handshaking" APM-08 interrupts.

MA2-MA0: These bits provide the current analog multiplexer channel address as follows:-

MA2	MA1	MA0	CHANNEL
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

3.6 THE APM-08 CONTROL REGISTER

The control register sets the multiplexer (channel) address, enables and disables interrupts and provides output data to the 4 general purpose digital outputs OP1-OP4. The control register is a write only register located at I/O address BASE + 2 (same location as status register). The data format of the control register is:-

BIT POSITION	D7	D6	D5	D4	D3	D2	D1	D0
(BASE + 2)	OP4	OP3	OP2	OP1	INTE	MA2	MA1	MA0

The bits have the following significance:-

OP4-OP1: These bits correspond to the four general purpose digital output lines OP1 thru OP4. These lines can be used for external control functions e.g. driving an input sub-multiplexer to increase the number of analog input channels. A 16 channel mux. on each of APM-08's 8 analog channels can expand the system to 128 channels.

INTE: APM-08 generated interrupts are enabled onto the common Apple interrupt bus. This bus uses a wire "OR" structure so that any peripheral board generating an interrupt will pull it active low. If there is more than one device generating interrupts, the user's interrupt service routine should first establish which device generated the interrupt. On the APM-08 the IRQ bit in the status register provides this information. It is cleared by writing to the control register. The interrupt service routine can be set up to perform many different functions e.g. background data acquisition, D/A waveform generation etc. but these capabilities are only available to the assembly language programmer. To disable interrupts, set INTE = 0.

MA2-MA0: These bits select the current analog multiplexer channel address as follows:-

MA2	MA1	MA0	CHANNEL
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

The multiplexer channel address can be determined at any time by reading the status register.

One further note about the control register. During power up of the Apple II when the RESET line is asserted, the APM-08 control register is cleared. This insures that APM-08 interrupts are disabled, sets digital outputs OP1-4 to zero and sets the multiplexer channel address to zero.

3.7 THE COUNTER TIMER REGISTERS

An 8254 programmable interval timer is used on APM-08. This is a very flexible device consisting of 3 separately programmable 16 bit down counters that may be operated in a variety of modes. A fuller description of the capabilities is in Chapter 4 (Counter Timer Operation).For additional technical information on this device, consult the "Intel Component Data Catalog"[1] or equivalent manufacturer's data sheet.

From a programming standpoint addressing counter timer functions is straightforward. The counter registers themselves are read write and located at addresses:-

BASE + 4 :	Counter 0
BASE + 5 :	Counter 1
BASE + 6 :	Counter 2

Before reading or writing to the counter registers, you should write to the counter timer control register to define the operating mode of each counter and the type of data transfer that you intend to make. The counter timer control register is write only and located at BASE + 7. It has the following format:-

BASE + 7 : 8254 Control (Write only)

BIT POSITION	D7	D6	D5	D4	D3	D2	D1	D0
(BASE + 7)	SC1	SC0	RL1	RL0	M2	M1	M0	BCD

SC1-0: These are the "select counter" bits that control which counter the following configuration bits will operate on. The format for the SC1-0 bits is:-

SC1	SC0	Addressed Counter
0	0	Counter 0
0	1	Counter 1
1	0	Counter 2
1	1	Read back command

RL1-0: These are the "read/load" configuration bits that control the form of the data transfer to the selected counter. The format for the RL1-0 bits is:-

RL1	RL0	Data Xfer Operation
0	0	Counter latching operation
0	1	Read/load high byte
1	0	Read/load low byte
1	1	Read/load low then high byte (2 byte transfer)

See Chapter 4 on Counter Timer Operation for a fuller description of these data transfer modes.

----

[1] Available from Intel Corporation, 3065 Bowers Avenue, Santa Clara, CA. 95051. Phone [408]-987-8080

M2-0: These are the selected counter operating mode control bits. Their format is:-

M2	M1	M0	Counter Mode
0	0	0	0 - Change on terminal count
0	0	1	1 - Programmable one-shot
0	1	0	2 - Rate generator
0	1	1	3 - Square wave generator
1	0	0	4 - Software triggered strobe
1	0	1	5 - Hardware triggered strobe

See Chapter 4 on Counter Timer Operation for a fuller description of these operating modes.

BCD: This bit controls whether the selected counter will count in binary or binary coded decimal (8,4,2,1 BCD) code.

BCD	Counting Code
0	16 bit binary (65,535 max.count)
1	4 decade BCD (9,999 max. count)

3.8 D/A DATA FORMAT

Data for the two D/A channels is left justified as follows:-

Most significant byte (hi byte):-

BIT POSITION	D7	D6	D5	D4	D3	D2	D1	D0
D/A 0: (BASE + 9) D/A 1: (BASE + 11)	B1 (MSB)	B2	B3	B4	B5	B6	B7	B8

Less significant nybble (lo byte):-

BIT POSITION	D7	D6	D5	D4	D3	D2	D1	D0
D/A 0: (BASE + 8) D/A 1: (BASE + 10)	B9	B10	B11	B12 (LSB)	x	x	x	x

x = don't care

The D/A converters are double buffered. This means that data written to the less significant byte is temporarily held in a special register in the D/A and combined with the high byte when it is written. The effect is to present the full 12 bits of data to the D/A at the same instant, and in this way avoid a two step change in the analog output. For some applications e.g. driving a plotter, a single step change is an important requirement. The only programming tradeoff is that it is impossible to change the 4 least significant bits of the D/A without writing to the 8 most significant bits. Usually this presents no difficulty.

Data for the D/A's is true binary. Digital zero input corresponds to zero output and digital full scale (12 bits = 4095) corresponds to full scale. When used with the fixed -10v reference input the scaling is as follows:-

BINARY	HEX	DECIMAL	ANALOG OUTPUT VOLTAGE
0000 0000 0000	000	0	0.0000 v (zero)
0000 0000 0001	001	1	0.0024 v (1 bit)
. . .	.	.	.
. . .	.	.	.
0100 0000 0000	400	1024	2.5000 v (1/4 scale)
. . .	.	.	.
. . .	.	.	.
1000 0000 0000	800	2048	5.0000 v (1/2 scale)
. . .	.	.	.
. . .	.	.	.
1100 0000 0000	C00	3072	7.5000 v (3/4 scale)
. . .	.	.	.
. . .	.	.	.
1111 1111 1111	FFF	4095	9.9976 v (Full scale)

An example procedure in BASIC to format and write data to D/A #1 from variable Y% (range 0-4095) is as follows:-

```
xxx10 YH% = INT(Y%/16)        :REM separate high byte
xxx20 YL% = 16*(Y%-16*YH%)    :REM separate low byte
xxx30 POKE BASE + 10, YL%     :REM write low byte
xxx40 POKE BASE + 11, YH%     :REM write high byte & load
```

3.9 USING THE ROM BASED DRIVER WITH APPLESOFT

The on-board driver ROM contains software for high level interface with Applesoft using the PR# & IN# functions. This saves a lot of PEEKing and POKEing to absolute addresses, data formatting etc. A typical example to perform an A/D conversion on channel 6 on an APM-08 in slot 3 and return the data to variable A% would be as follows:-

```
xxx10 PR#3 : IN#3            :REM I/O to slot 3
xxx20 PRINT "C6"             :REM Send convert command
xxx30 INPUT A%               :REM Return data
xxx40 PRINT "Q"              :REM Return control to DOS
```

If you are not using DOS 3.3, then instead of line xxx40 substitute:-

```
        xxx40 PR#0 : IN#0           :REM Return control to
                                     screen & keyboard.
```

The price paid for this simplicity is a loss in speed due to the relatively slow execution of interpreted BASIC and the loss of some of the hardware capabilities e.g. no interrupt capabilities supported by BASIC. However, for many less speed conscious applications using the ROM driver saves a lot of programming time and complexity. A source listing for the driver ROM is contained in Appendix E.

The command set for the ROM driver is as follows:-

COMMAND	+	DATA	FUNCTION
C		0 - 7	Performs A/D conversion on channel 0 - 7.
D0		0 - 4095	Output data to D/A #0
D1		0 - 4095	Output data to D/A #1
S0		0 - 5	Set counter 0 configuration
S1		0 - 5	Set counter 1 configuration
S2		0 - 5	Set counter 2 configuration
L0		-	Latch counter 0
L1		-	Latch counter 1
L2		-	Latch counter 2
O		0 - 15	Write to 4 bit output port.
W0		0 - 65535	Load counter 0.
W1		0 - 65535	Load counter 1.
W2		0 - 65535	Load counter 2.
I		-	Read 3 bit input port.
R0		-	Read counter 0.
R1		-	Read counter 1.
R2		-	Read counter 2.
Q		-	Quit to DOS (last command)

Remember that the PRINT function always outputs data as an ASCII string to the peripheral board. Several constructions are possible e.g.:-

```
1:        xxx10   PRINT "C4"

2:        yyy10   I% = 4
          yyy20   PRINT "C";I%

3:        zzz10   A$ = "C"
          zzz20   I% = 4
          zzz30   PRINT A$;I%
```

These are all equivalent. Using a variable in the PRINT opens up several possibilities. As an example, consider the following routine for scanning all channels into an array:-

```
xxx10  DIM D%(7)          :REM data array
xxx20  PR#3 : IN#3        :REM I/O to slot 3
xxx30  FOR I% = 0 TO 7
xxx40  PRINT "C";I%
xxx50  INPUT D%(I%)
xxx60  NEXT I%
xxx70  PRINT "Q"          :REM return control
```

Note that commands must be output as a continuous string. Spaces are ignored, but tabs and returns are not, so that statements such as the following are illegal:-

```
zzz10 PRINT "C",A%

yyy10 PRINT "D" : PRINT "0"
```

Chapter 4

COUNTER/TIMER

4.1 THE 8253 PROGRAMMABLE INTERVAL TIMER

The Intel 8254 programmable interval timer is used in the APM-08. This is a flexible but somewhat complex device consisting of three independent 16 bit pre-settable down counters. The main uses of the 8254 are:-

1. - A programmable timer for generating interrupts and triggering periodic A/D conversions.

2. - A variable frequency square wave generator for testing and frequency synthesis.

3. - An event counter for external pulse inputs.

4. - A time delay generator.

In addition, it is possible to accurately measure frequency and period by interconnecting some of the counters. For those interested in detailed information, a full description of the 8254 programmable interval timer can be found in the Intel data sheet (or equivalent manufacturer's literature).

Each counter has a clock input, a gate input that controls counting and triggering and an output. The maximum clock input frequency on any counter is 8MHz with minimum clock duty cycles of 60 nS high and 60nS low. A block diagram of the 8254 configuration in APM-08 is detailed in Fig. 4.1. There are 5 possible operating configurations for each counter:-

Configuration	Description
0	PULSE ON TERMINAL COUNT. The output is initially low after setting this configuration. After the count is loaded, the output remains low until the counter decrements thru zero, when it goes

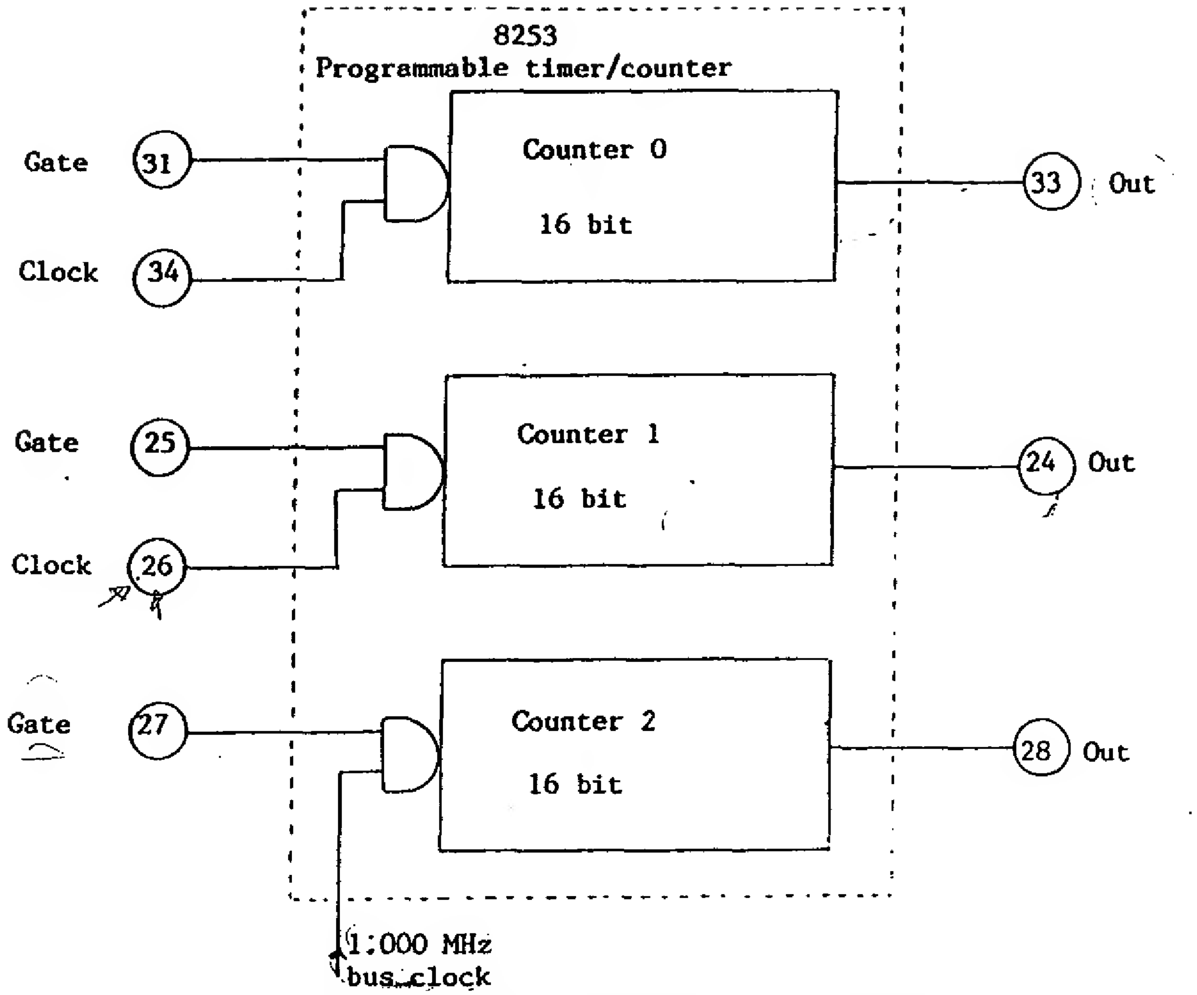


Fig. 4.1 8253 TIMER/COUNTER CONFIGURATION

high and remains high until the counter is reloaded. The counter will continue to decrement after passing thru zero and counting can be inhibited by a low gate input. This mode produces a single positive going output transition such as may be required in a time delay initiated by the program.

1 PROGRAMMABLE ONE SHOT. The output goes low after a rising edge of the gate input and goes high when the counter passes thru zero. The period that the output is low is set by the loaded count. If the gate input goes high again before the one shot has timed out, a new timing cycle is initiated i.e. the one shot is re-triggerable and if a new count is loaded, it will not become effective until any cycle in progress has terminated. This provides a hardware triggered delay or one-shot.

2 RATE GENERATOR (or divide by N counter). The output goes low for one input clock period every N counts, where N is the count loaded. The gate input when low, forces the output high, and on going high, reloads the counter. Thus the gate input can be used to synchronize the counter. This configuration is useful for generating periodic interrupts to trigger A/D conversions.

3 SQUARE WAVE GENERATOR. This is similar to configuration 2 except that the output is high for half of the count and low for the other half. If N is even, a symmetrical square wave output is obtained. If N is odd, the output is high for (N+1)/2 counts and low for (N-1)/2 counts i.e. has a 1 count assymmetry. This configuration can be used in the same way as configuration 2 for periodic triggering or for frequency synthesis.

4 SOFTWARE TRIGGERED STROBE. After the mode is set the output is high. When a count of N is loaded the counter begins counting, and the the output will go low for one input clock period as it passes thru zero. The cycle is repeated on loading another count. The gate input may be used to inhibit counting.

5 HARDWARE TRIGGERED STROBE. This is essentially the same as configuration 1, except that the output will go low for one clock period at the end of the cycle and return high again. The start of the cycle is triggered by the rising edge of the gate input, and as in configuration 1, is retriggerable.

The 8254 programmable interval counter uses 4 memory address locations:-

Address	Register type	Description
BASE + 4	Read/write	Counter 0
BASE + 5	Read/write	Counter 1
BASE + 6	Read/write	Counter 2
BASE + 7	Write only	Control

Before loading or reading any of the individual counters, the control register must be loaded with data setting the counter operating configuration as above, the type of read or write operation that will be performed (see following) and the modulus, binary (0 -

65,535) or BCD (Binary coded decimal 0 - 9,999). The format of the control byte is:-

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	RL1	RL0	M2	M1	M0	BCD

SC1-0 - Control which counter is selected.

SC1	SC0	Counter
0	0	0
0	1	1
1	0	2
1	1	Status readback (see data sheet)

RL1-0 - Control the type of read/load operation.

RL1	RL0	Operation
0	0	Counter latch (see following)
0	1	Read/load most significant byte
1	0	Read/load least significant byte
1	1	Read/load least significant byte, followed by most significant byte.

M2-0 - Control counter configuration as above.

M2	M1	M0	Configuration
0	0	0	0 - Pulse on terminal count
0	0	1	1 - Programmable one shot
X	1	0	2 - Rate generator
X	1	1	3 - Square wave generator
1	0	0	4 - Software triggered strobe
1	0	1	5 - Hardware triggered strobe

BCD - Controls binary/decimal counting.

BCD	Counter type
0	Binary 16 bits
1	Decimal 4 decades

For each counter you are required to specify in advance the type of read or load operation that you intend to perform. You have a choice of loading/reading the high byte of the count or the low byte of the count, or the low byte followed by the high byte. This last mode is of the most general use and is selected for each counter by setting the RL 1/0 bits to "1 1". Subsequent read/load operations must be performed in pairs in this sequence, otherwise the internal sequencing flip-flop of the 8254 will get out of step. With RL0 & RL1 will both be set to 1 to perform lo byte/high byte reads or loads, the following example shows how data is loaded in the correct sequence e.g. to load 30,000 into counter 0:-

```
        xxx10 XH% = INT(30000/256)       :REM Calculate hi byte
        xxx20 XL% = 30000 - XH%*256      :REM Calcualate low byte
        xxx30 POKE BASE + 4, XL%         :REM Load low byte
        xxx40 POKE BASE + 4, XH%         :REM Load hi byte
```

Note how both bytes are loaded sequentially into the same address to load the full 16 bits of data.

If you attempt to read the counters on the fly with a high input frequency, you will most likely obtain erroneous data. This is partly caused by the rippling of the counter during the read and also by the fact that the low and high bytes are read sequentially rather than simultaneously, making it highly probable that carries will be propagated from the low to high byte during the read cycle. To circumvent these problems, you can perform a counter latch operation in advance of the read cycle. To do this you load the RL 1/0 bits of the control byte with "0 0" which instantaneously latches the count of the selected counter in a 16 bit latch register. A subsequent read operation on the selected counter returns the contents of the latch. This is the only satisfactory way of reading a counter on the fly without discontinuing the counting process.

The counters may be programmed to count in binary (modulus 2) or binary coded decimal (modulus 10) modes by the BCD bit. The binary mode with a full count of 65,535 has the obvious advantage of providing a larger count range than the BCD mode which has a 9,999 full scale.

4.2 EVENT COUNTING OR COUNTING A NUMBER OF INPUT PULSES

One of the common applications for the 8254 is counting pulses or events. Pulses should be clean TTL signals or de-bounced signals from contact closures and should be connected to the clock input of the selected counter. The corresponding gate input can be used to enable and disable counting operations. Only Counters 0 and 1 have external clock inputs on APM-08, so these should be used for pulse counting. Counter 2 clock input is internally connected to the computer bus clock and this counter is limited to time interval generation and frequency synthesis.

Configuration 0 is a good choice for plain counting. Assuming the use of counter 0, first set the control register to select configuration 0 with a lo byte/hi byte load sequence. This corresponds to a control word of 30 Hex (or 48 decimal):-

```
          xxx10 POKE BASE + 7,48       :REM Set control byte
```

Next, since the counter will always count down, it should initially be loaded with a full scale count (65,535) or at least a value that will exceedd the anticipated count total. Loading with 65,535 corresponds to a low byte of 255 and a high byte of 255:-

```
        xxx20 POKE BASE + 4,255      :REM Lo byte
        xxx30 POKE BASE + 4,255      :REM Hi byte
```

Now the counter is initialized, the gate input can be taken high to commence counting. The gate could be controlled externally or connected to one of the APM-08 digital outputs and controlled by software or simply left open circuit to continuously enable the counter.

The counter can be read in 2 ways. If pulses are no applied or the gate input taken low to disable the counter, then an ordinary non-latched read can be performed:-

```
        xxx40 XL% = PEEK(BASE + 4)    :REM Read low byte
        xxx50 XH% = PEEK(BASE + 4)    :REM Read high byte
        xxx60 COUNT = 65535 - 256*XH% - XL%
```

Note line xxx60 where the change in count is calculated. This is a necessary step for a down counter.
If we wished to read the counter "on the fly" without disabling it or altering the count, then a counter latch operation should be performed before reading:-

```
        xxx40 POKE BASE + 7,0         :REM Latch counter 0
        xxx50 XL% = PEEK(BASE + 4)    :REM Read low byte
        xxx60 XH% = PEEK(BASE + 4)    :REM Read high byte
        xxx70 COUNT = 65535 - 256*XH% - XL%
```

4.3 GENERATING SQUARE WAVES OF PROGRAMMED FREQUENCY

Counter 2 clock input is connected internally on the APM-08 board to a 1.0 Mhz. input signal derived from the main computer clock. Counter 2 can be operated in configuration 3 (square wave generator) with a maximum divisor of 65,535. The lowest output frequency obtainable from Counter 2 directly will be 15.3 Hz (1000000 / 65535). The minimum divisor can be 2 to obtain a maximum output frequency of 500 Khz. Frequencies lower than 15.3 Hz are easily obtained by cascading the output of Counter 2 into the clock input of Counter 0 or 1. Obviously a further division by 65,535 would yield a very low frequency (1 cycle per hour).

In practice, to obtain a symmetrical square wave, the divisor loaded into the counter should be an even number. If it is an odd number, one half of the square wave will be 1 clock pulse (1.0 microsec) longer than the other half.

Calculating the divisor is straightforward. Assume you desire an output frequency of 1 KHz. The input frequency to the counter is 1 Mhz so you must divide this by 1000 to obtain 1 Khz. Counter 2 should be set in configuration 3 and loaded with 1000 as follows:-

```
xxx10 POKE BASE + 7, 182   :REM Hex B6 control byte
xxx20 XH% = INT(1000/256)  :REM Calculate hi byte
xxx30 XL% = 1000 - 256*XH% :REM Calculate lo byte
xxx40 POKE BASE + 6, XL%   :REM Load lo byte
xxx50 POKE BASE + 6, XH%   :REM Load hi byte
```

Counter 2 output will now be a 1KHz square wave.

4.4 MEASURING FREQUENCY AND PERIOD

The two previous sections show how to count pulses and output frequencies. It is possible to use the 8254 to measure frequency by raising the gate input of a counter for some known interval of time, say 10, 100 or 1000mS and counting the number of pulses clocked into the counter for that interval. The gating signal can be derived from counter 2 and a second cascaded counter both operating in square wave mode. Also the computer has to be informed about the start and finish of the measurement cycle, so one of the APM-08 digital inputs can be used to monitor the gate input to achieve this requirement.

Counter 2 can be used to measure pulse width or half period of a periodic signal. The signal should be applied to the gate input of Counter 2. During the interval when the gate input is low, Counter 2 is loaded with a full count, 65,535. The gate input then goes high at the beginning of the measurement, and the counter decrements until the gate input goes low at the end of the measurement. The counter is then read and the change in the count is the duration of the gate input signal. Since Counter 2 input is fed with 1 microsecond duration clock pulses (1 MHz), the maximum pulse duration that can be measured using Counter 2 alone is 65.5 milliseconds. Longer pulse durations can be measured using another counter and driving its input from a known frequency derived from the output of Counter 2.

65.5 mS

4.5 GENERATING TIME DELAYS

Another use for the programmable interval timer is generating accurate time delays. There are several "one shot" modes that the counters can be configured in. The counter configurations have the following characteristics when used for time delay generation:-

Configuration 0 Pulse on terminal count. - After loading the counter the output goes low. Counting is enabled when the gate input is high and continues until the count reaches zero when the output goes high. The output will remain high until the counter is reloaded by a programmed command. Taking the gate input low during the count down will disable counting as long as it is low.

Configuration 1 Programmable one shot. - The counter need only be loaded once. The timing delay is initiated by the gate input going high. At this point the output goes low. If the gate input goes low, counting continues but a new cycle will be initiated if the gate input goes high again before the time out delay has expired i.e. is re-triggerable. At the end of the time out, as the counter reaches zero, the output goes high and will remain high until re-triggered by the gate input. This is the programmable equivalent of a "one shot" or monostable, hence the name.

Configuration 4 Software Triggered Strobe. - This is similar to configuration 0, except that after loading the output goes high and only goes low for one clock period on timing out. This produces a negative strobe pulse a programmed duration after loading the counter.

Configuration 5 Hardware Triggered Strobe. - This is similar to configuration 1, except that the time out is triggered by the gate input going high and the output is normally high, going low for one clock period on time out and producing a negative going strobe pulse. Like configuration 1, the time out is re-triggerable i.e. a new cycle will commence if the gate input is taken high before a current cycle has timed out.

Counter 2 is good for directly producing delays up to 65.5mS. For longer delays, Counter 2 should be operated in the square wave mode to output a suitable frequency to feed into one of the other counters set up in one of the programmable delay configurations. In theory, using all the counters, a delay as long as 65,535 * 65,535 * 65,535 microseconds, or about 9 years, can be produced this way, although obviously this is of academic interest only!

4.6 TRIGGERING THE A/D PERIODICALLY

Another of the key uses for the 8254 programmable interval timer is in providing trigger pulses for an interrupt service routine that performs periodic A/D conversions. Actually the APM-08 hardware can be used to trigger any interrupt service routine from the counter or an external input and is not limited to servicing the A/D. The output of counter 2 should be connected to the interrupt input (pin 19) to generate timer interrupts. It is also necessary to set the INTE enable bit in the APM-08 control register and for your interrupt service routine to write to the control register each time to clear the IRQ flip flop.

Chapter 5

APPLICATIONS

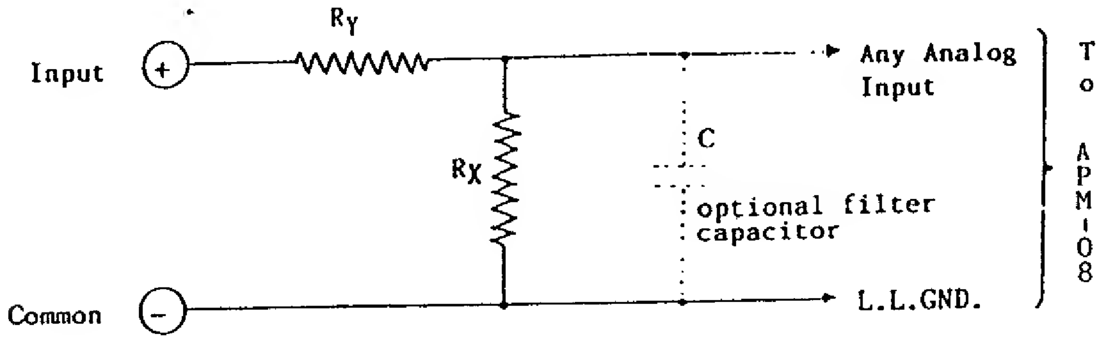
5.1 CHANNEL INPUTS

There are 8 analog input channels on APM-08. Each has an input range of -5.000v to +4.9976v and are single ended i.e. they share a common low level ground. Input voltages should be applied between the channel Hi and any L.L. Gnd. Do not return inputs to the digital common (DIG.COM.) as this is intended as a heavy current return for power supplies and digital logic signals and may differ from the low level ground by many millivolts. Correct use of the grounds is very important to obtain consistent noise free measurements as it is easy to introduce inadvertent ground loops when using single ended connections. The low level grounds are used for all analog signal returns and when used correctly should only carry signal currents less than a few milliamps. The seven identical low level ground inputs have been positioned in the connector so that they lie between the analog channel inputs in the flat connecting cable, this helps to prevent crosstalk. The input current of each channel is about 100 nanoamps at 25 deg. C. thus presenting a high input impedance to the signal. Also the 508A solid state channel multiplexer used on the APM-08 is designed to withstand continuous overloads of +/- 32v on each channel and transient overloads of several hundred volts. This multiplexer has two other desirable characteristics, a "break before make" action to prevent shorts between channels while switching, and all channel switches turn off when the power is off thus preventing signal to signal shorts when your computer is off.

5.2 MEASURING VOLTAGE

Voltages in the range +/-5v may be directly applied to the analog inputs. Higher voltages should be attenuated, a simple resistive divider should be adequate as shown in Fig. 5.1.

Single ended inputs have a common ground return which is connected to the ground (case) of the computer. If you are measuring a signal which is floating i.e. has no connection to ground, there will be no problem but if your signal source is also connected to ground, then there is the potential for a ground loop which may cause an error or noise in your readings. There are several ways to avoid this complication, some of the solutions are shown in Figs. 5.2, 5.3 & 5.4. All of these méthods provide you with a differential input



Attenuation = $(R_X + R_Y) / R_X$

Typical Values

Attenuation ratio	R_Y	R_X
x 2	10K	10K
x 10	90K	10K
x 100	99K	1K
x 1000	999K	1K

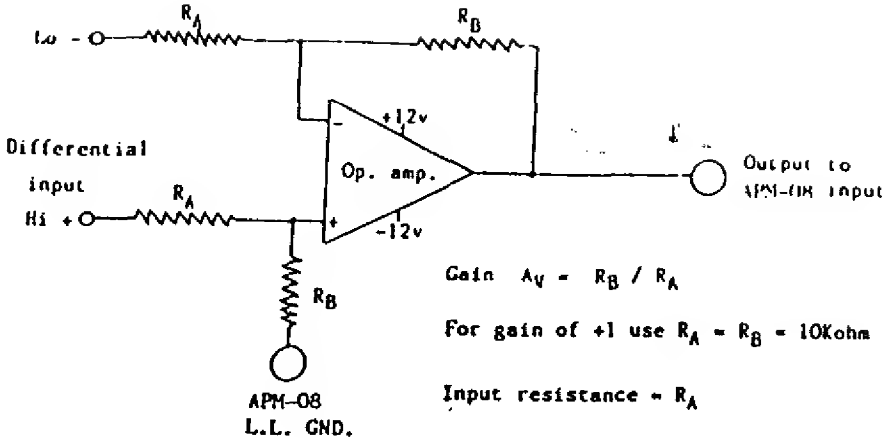
Fig. 5.1 SIMPLE ATTENUATOR FOR VOLTAGES GREATER THAN +/-5v.

which allows you to reject any small differences in ground potential between your computer and signal source.

The circuit of Fig. 5.2 is the least expensive, but has the draw back of having an input resistance set by the input resistors. This may be quite large, in the 10Kohm to 100Kohm region, but may be too low for some applications. As an added benefit, the resistors may be chosen to provide gain or attenuation. This circuit is the classic differential connection for an operational amplifier and a full description can be found in any book on Operational Amplifiers[2].

----

[2] See for instance "Operational Amplifiers - Design and Applications" by Tobey, Graeme & Huelsman. McGraw-Hill 1971.

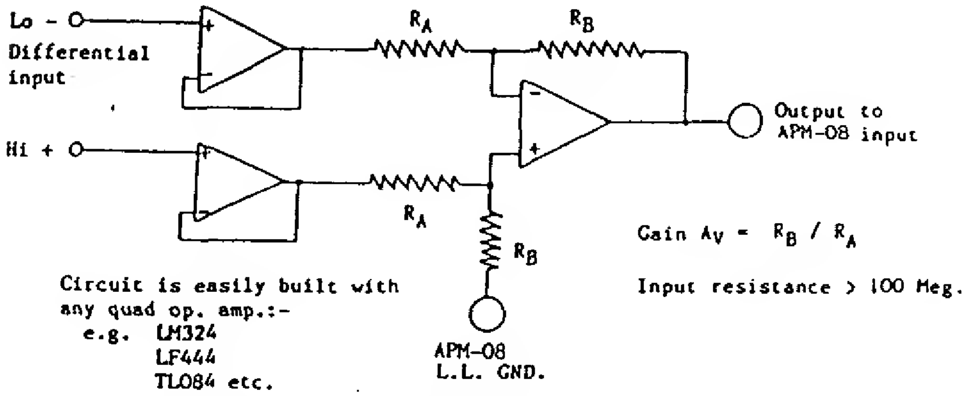


Gain $A_V = R_B / R_A$

For gain of +1 use $R_A = R_B = 10\text{Kohm}$

Input resistance = R_A

Fig. 5.2 SIMPLE DIFFERENTIAL AMPLIFIER USING AN OPERATIONAL AMPLIFIER

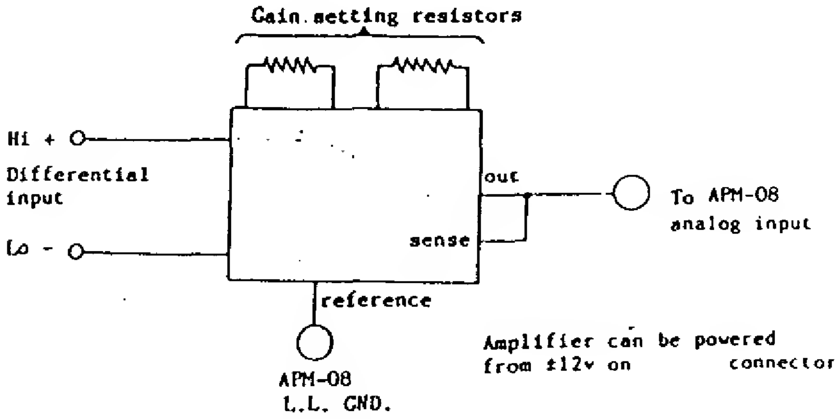


Gain $A_V = R_B / R_A$

Input resistance > 100 Meg.

Circuit is easily built with
any quad op. amp.:-
e.g. LM324
LF444
TL084 etc.

Fig. 5.3 ADDING BUFFERS TO CIRCUIT OF Fig. 5.2 FOR HIGH IMPEDANCE INPUT



Amplifier can be powered
from ±12v on connector

Typical instrumentation amplifiers :-

LM363	-	National Semiconductor
AMP-01	-	Precision Monolithics Inc. (P.M.I.)
AD-524	-	Analog Devices

Fig. 5.4 CORRECT CONNECTIONS FOR AN INSTRUMENTATION AMPLIFIER

Fig 5.3 is a variant of the circuit of Fig 5.2 and adds two voltage followers to this circuit to provide a very high input impedance for sensitive signals. Finally if you want to buy a ready made differential amplifier, this part is available from integrated circuit manufacturers as a single component. In this form it is called an instrumentation amplifier, some types include gain setting resistors and others require external resistors. Instrumentation amplifiers are usually optimised for operation at high gains with small signals and usually have zero drifts of less than a few millionths (microvolts) per degree C.. Although more costly than simple operational amplifiers, operation under high gain conditions usually demands the extra stability and common mode rejection that instrumentation amplifiers provide.

These various methods provide a variety of different interfacing solutions of different costs and complexities. Almost certainly, one of these will be appropriate for your requirements.

All of these circuits can be conveniently mounted on the breadboard area of the STA-AP screw connector board. This area is provided with +/-12v power from the computer which in most cases will be adequate to power any interface circuitry.

5.3 4-20mA CURRENT LOOPS

Process control current loop transducers are easily interfaced to APM-08 by adding a suitable shunt resistor across the input. Since the maximum current will be 20mA and the maximum input range is +5v, a 250 ohm precision shunt resistor will be required. This should be of low temperature coefficient metal film or wirewound construction for stability with time and temperature.

Using this interface, the 4-20mA working range of the current loop corresponds to 1638 bits of input, a resolution of about 0.06%.

5.4 THE REFERENCE

A -10v stable voltage reference (-Vref) derived from the A/D reference is brought out for users. It may be used for offsetting signals etc. but should not be heavily loaded. The maximum available output current is 2mA. Since this reference may be used by the D/A's, any overload or shorting of the reference will affect their operation.

5.5 D/A CONVERTERES

The equivalent circuit of the D/A converters is shown below:-

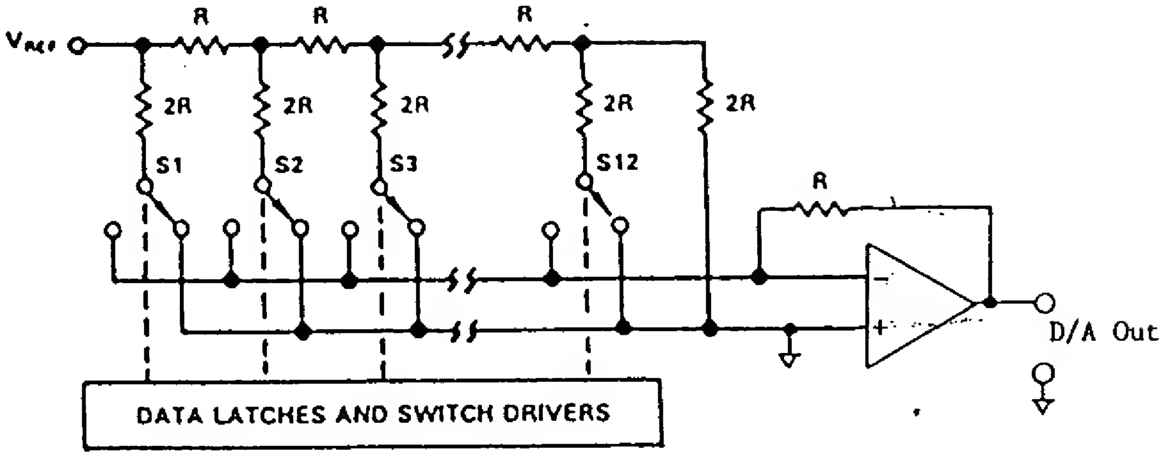


Fig. 5.5 Equivalent Circuit of D/A Channel

The D/A's are of the R/2R CMOS multiplying type. The transfer equation is:-

Vout = -Vref * (Digital data)/4096

The reference voltage Vref may range from -10 to +10v. To operate as a normal fixed range output D/A with a scaling of 0 to +10v a D.C. reference input of -10v is required. This is supplied on pin 16 of the APM-08 output connector and should be jumpered across to either or both of the selected D/A inputs (pins 22 or 23). Other scalings can be obtained with different reference voltages e.g. -5v reference would give 0 to +5v output, +2v reference would give 0 to -2v output etc.

It is also possible to use an A.C. reference and in this case the customary terminology of operation is somewhat different. 2 quadrant multiplication is obtained since the reference which may be positive or negative is multiplied with a positive only digital signal.

Two other parameters are of interest in A.C. operation. The first is feedthrough, the amount of residual signal at digital zero. The feedthrough which is mainly a function of stray capacitance rises with frequency. At 10KHz it is typically 5mV peak

- peak with a +/-5v reference. The second parameter that is a limit at a lower frequency, is the accuracy/frequency characteristic. Due to distributed capacitance in the R-2R ladder network, the full 12 bit performance of the D/A falls off as the frequency rises. Above about 1KHz the dynamic performance of the D/A will be less than 12 bit accurate.

The APM-08 D/A's will perform well in synchro-digital and resolver applications for sine/cosine generation with 400 Hz reference.

5.6 USING DIGITAL INPUT/OUTPUT

APM-08 provides 4 TTL/DTL compatible digital outputs (OP1-4) and 3 TTL/DTL compatible digital inputs.

The digital outputs correspond to bits 4 - 7 of the control register and are accessed by writing to the control register. When you write to the control register you will often need to maintain the state of bits 0 - 3 that control the multiplexer address and interrupt enable. For this reason, it is a good practice to store the control register byte in a variable e.g. C% so that digital I/O can be OR'ed with C% before loading the control register. This avoids disturbing the the other bits in the register during a digital output operation.

Digital outputs can sink 8mA (5 standard TTL loads or 20 LSTTL loads). If you wish to interface to CMOS, 1Kohm pull-up resistors connected to +5v should be attached to the outputs. This will raise the logic high output level from its minimum TTL level of 2.4v to +5v suitable for CMOS interface.

Digital inputs are available through bits 4 - 6 of the status register. The digital data is readily obtained by masking out these bits using a logical AND operation. The inputs present a -0.4mA loading corresponding to 1 LSTTL load.

5.7 PROGRAMMABLE INTERVAL TIMER

The 8254 interval timer provides time delays, counting, frequency synthesis and in compound configurations, frequency and period measurement. Since there are 3 independent counters, many ingenious applications and configurations are possible for this device. A full discussion of the capabilities is in Chapter 4.

5.8 ADDING MORE ANALOG INPUTS

You may add sub-multiplexers to any or all of the 8 analog inputs. MetraByte's EXP-16 provides 16 channels per input. Up to 8 EXP 16's can be added to one APM-08 providing a total of 128 channels. The sub-multiplexer address can be set by digital outputs OP1-4. The EXP-16 cards are designed to cascade with flat cable and insulation displacement connectors. All analog channel connections are made by screw connectors, and each EXP-16 (group of 16 channels) can be operated at a different gain. In this way a system can be configured with a variety of different channel functions and gains, single ended and differential.

5.9 INTERFACE TO TRANSDUCERS, THERMOCOUPLES ETC.

Low level transducers such as thermocouples and strain gage bridges (load cells, pressure & force transducers) require amplification before applying to the high level APM-08 inputs. The EXP-16 expansion multiplexer incorporates an instrumentation amplifier that can provide stable amplification and also includes circuitry that allows cold junction compensation of thermocouples. EXP-16 will handle most interfacing requirements to D.C. output transducers and also includes spaces for filters, shunts and attenuators.

For inexpensive temperature measurement in the -50 to +125 deg. C. temperature range, semiconductor temperature transducers are a good choice. The most popular types are the AD590 (Analog Devices) which behaves like a constant current source with an output of 273uA at 0 deg.C. and a scaling of 1uA/deg.C. and the LM335 (National Semiconductor) that has an output of 2.73 volts at 0 deg.C. and and a temperature coefficient of 10mV/deg.C.. Both of these devices can be powered from the +12v available from the computer and directly interfaced to APM-08.

For measuring high temperatures, up to 1800 deg.C. or more, thermocouples are the most satisfactory solution. The base metal thermocouples, types J,K,T & E, have outputs around 40 microvolts/deg.C., while the platinum and tungsten types used for the highest temperature measurement, types S,B,& R, tend to have lower outputs in the 6-12 microvolt/deg. C. range. A further complication encountered in the use of thermocouples is the "cold-junction" compensation. Where the thermocouple wire is terminated to the copper APM-08 connections, an unwanted thermocouple junction is formed. As the connector temperature varies, this introduces an error. The error can be bucked out by sensing the connector temperature using a semiconductor sensor on another channel, and correcting the thermocouple readings in software. This is only required at the highest levels of accuracy, since in most cases connector temperature (usually room temperature) varies little.

5.10 POWER OUTPUT FROM THE APM-08 CONNECTOR

The +5v and +/-12v Apple power supplies are available on the APM-08 rear connector. These are provided as a convenience to users who wish to add external signal conditioning and logic circuits. The +/-12v can be used for analog circuits, operational amplifiers, comparators, indicators relays etc. and the +5v will power logic circuits, TTL, CMOS etc. Careful use of these supplies can often avoid the expense and bulk of external supplies to power your signal sources and the nuisance of multiple power sources and switches. If you intend to use these supplies observe the loading limits. The amount of power available is limited and depends to a considerable extent on what other peripheral boards are plugged into your Apple. In most cases there will usually be adequate power for analog circuits which consume a few tens of milliamps and a few 74LS TTL or many CMOS logic circuits.

If the power outputs are subjected to an overcurrent (overload) or overvoltage condition, the power supply is designed to shut down and the computer may have to be turned off and turned on again to restore normal operation after removing the fault. Although protective devices are built into your computer supply, use your computer power with care and consideration. This convenience is not to be abused, so if there is any possibility of frequent short circuits or shorts to high voltages and signal sources, then it is advisable to provide an external (and more easily repaired) power supply for your user circuits.

5.11 PRECAUTIONS IN USE - NOISE, GROUNDLOOPS AND OVERLOADS

Unavoidably, data acquisition systems give users access to inputs to the computer. Do NOT, whatever else you do, get these inputs mixed up with the A.C. line. An inadvertent short can in an instant can cause extensive and costly damage to your computer. MetraByte can accept no liability for this type of accident. As an aid to avoiding this problem:-

1. - Avoid direct connections to the A.C. line.

2. - Make sure that all connections are tight and sound

so that signal wires are not likely to come loose and short to high voltages.

3. - Use isolation amplifiers and transformers where necessary.

There are two ground connections on the rear connector called DIG. COM. and L.L. GND. Digital common is the noisy or "dirty" ground that is meant to carry all digital signal and heavy current (power supply) currents. Low level ground is the signal ground for all analog input functions. It is only meant to carry signal currents (less than a few mA) and is the ground reference for the A/D channels. Due to connector contact resistance and cable resistance there may be many millivolts difference between the two grounds although they are connected to each other and the computer and power line grounds on the APM-08 board.

Appendix A

CONNECTIONS

A.1 MAIN I/O CONNECTOR

The main analog and digital I/O is via a 40 pin header type connector at the rear of the board. The pin functions are as follows (see Fig A.1 for locations):-

PIN	NAME	FUNCTION
1	IN 7	Channel 7 analog input
2	L.L.GND.	Low level ground
3	IN 6	Channel 6 analog input
4	L.L.GND.	Low level ground
5	IN 5	Channel 5 analog input
6	L.L.GND.	Low level ground
7	IN 4	Channel 4 analog input
8	L.L.GND.	Low level ground
9	IN 3	Channel 3 analog input
10	L.L.GND.	Low level ground
11	IN 2	Channel 2 analog input
12	L.L.GND.	Low level ground
13	IN 1	Channel 1 analog input
14	L.L.GND.	Low level ground
15	IN 0	Channel 0 analog input
16	-Vref	-10 v reference voltage

17	D/A 1	D/A #1 analog output
18	D/A 0	D/A #0 analog output
19	INT.IN	Interrupt input. Positive edge triggered.
20	+12v	+12v power supply from Apple II.
21	+5v	+5v power supply from Apple II.
22	REF.IN#0	Reference input for D/A #0
23	REF.IN#1	Reference input for D/A #1
24	CTR.1 OUT	8253 Counter 1 output
25	GATE 1	8253 Counter 1 gate
26	CLK. 1	8253 Counter 1 clock input
27	GATE 2	8253 Counter 2 gate
28	CTR.2 OUT	8253 Counter 2 output
29	OP2	Digital output #2
30	OP3	Digital output #3
31	GATE 0	8253 Counter 0 gate
32	-12v	-12v power supply from Apple computer
33	CTR.0 OUT	8253 Counter 0 output
34	CLK 0	8253 Counter 0 clock input
35	OP1	Digital output #1
36	DIG. COM.	Digital common. Return for all logic signals and power supply currents. Connected to frame & line ground.
37	OP0	Digital output #0
38	IP0	Digital input #0
39	IP1	Digital input #1
40	IP2	Digital input #2

The insulation displacement (flat cable) mating connector for the 40 pin header is 3M part number 3595 - 6002. Similar types are available from several other manufacturers. For users not intending to make flat cable connector to connector interconnects, use of the STA-AP screw connector board is recommended.

A.2 REAR VIEW OF APM-08 CONNECTOR

IN 7	1	2	L.L. GND.
IN 6	3	4	L.L. GND.
IN 5	5	6	L.L. GND.
IN 4	7	8	L.L. GND.
IN 3	9	10	L.L. GND.
IN 2	11	12	L.L. GND.
IN 1	13	14	L.L. GND.
IN 0	15	16	-Vref (-10v)
D/A #1 OUT	17	18	D/A #0 OUT
INT. IN	19	20	+12v power
+5v power	21	22	REF. IN# 0
REF. IN# 1	23	24	CTR. 1 OUT
GATE 1	25	26	CLK. 1
GATE 2	27	28	CTR. 2 OUT
OP2	29	30	OP3
GATE 0	31	32	-12v power
CTR. 0 OUT	33	34	CLK. 0
OP1	35	36	DIG. COM.
OP0	37	38	IP0
IP1	39	40	IP2

Appendix B

SPECIFICATIONS

B.1 POWER CONSUMPTION

+5v supply	-	295mA typ. / 320mA max.
+12v supply	-	25mA typ. / 35mA max.
-12v supply	-	23mA typ. / 30mA max.

B.2 A/D SPECIFICATION

Type	-	Successive approximation with sample/hold.
Conversion time.	-	25 microseconds typ. 35 microseconds max.
Monotonicity	-	Guaranteed over operating temperature range.
Linearity	-	+/-1 bit.
Resolution	-	12 bits. (2.4mV/bit)
Accuracy	-	0.01% of reading +/-1 bit.
Full scale	-	+/-5 volts
Coding	-	Offset binary
Overvoltage	-	Continuous single channel to +/-35v
Configuration	-	Single ended.
Input current	-	100nA max at 25 deg.C.
Zero drift	-	10ppm/deg. C. max.

Gain drift	-	50 ppm/deg. C max. (30 ppm/deg C. available to special order)

B.3 SAMPLE HOLD AMPLIFIER

Acquisition time	-	15 microsecs to 0.01% typ. for full scale step input
Dynamic sampling error	-	1 bit (2.44mV) @ 2000v/sec.

B.4 REFERENCE VOLTAGE OUTPUT

Reference voltage	-	-10.0v +/- 0.1v
Temperature coefficient	-	50 ppm/deg.C max. (30 ppm/deg.C available to special ordwer)
Load current	-	+/-2mA max.

B.5 D/A CONVERTERS

Channels	-	2
Resolution	-	12 bits (1 part in 4095)
Relative accuracy	-	+/-1/2 LSB (0.01%) max.
Differential linearity	-	1/2 LSB max.
Fixed reference output range (using -10v ref.)	-	0 to +10v
Variable reference output range	-	+/-10v
Reference input	-	7Kohm min., 10 Kohm typ., 20Kohm max.

resistance

Voltage output resistance	-	< 0.1 ohm max.
Output drive current	-	+/-5mA min.
Settling time to 0.01% for F.S. step.	-	35 microsecs max.
Gain temperature coefficient	-	+/-10ppM/deg.C. max (excluding reference)

B.6 DIGITAL I/O

OP1-4 output low voltage	-	0.5v max at Isink = 8.0mA
OP1-4 output high voltage	-	2.7v min at Isource = -0.4mA
IP1-3 input low voltage	-	0.8v max
IP1-3 input low current	-	-0.4mA max
IP1-3 input high voltage	-	2.0v min
IP1-3 input current	-	20uA max. @ 2.7v

B.7 INTERRUPT INPUTS

Type	-	Positive edge triggered
Enable	-	Via INTE of CONTROL register

Interrupts are latched in an internal flip-flop on the APM-08 board. The state of this flip-flop corresponds to the IRQ bit in the STATUS register.

Flip-flop is cleared by a write to the CONTROL register. Service routines should acknowledge and re-enable interrupt flop.

B.8 COUNTER/TIMER

Type	-	8254 programmable interval timer
Counters	-	3 down counters, 16 bit.
Output drive capability	-	2.2mA @ 0.45v (5 LSTTL loads)
Input, gate & clock load	-	TTL/DTL/CMOS compatible +/-10 uA current
Max. input clock freq.	-	Not less than 8 MHz (may vary with manufacturer)
Active count edge	-	Negative
Minimum clock pulse widths	-	60nS high / 60nS low

For additional information on programming see Chapter 4.

B.9 POWER OUTPUTS

Apple buss supplies	-	+5v & +/-12v
Tolerance	-	+5v +/-5% +12v +/-5% -12v +/-10%
Loading	-	Dependent on other peripherals (see Apple Tech. Ref. Manual)

B.10 GENERAL ENVIRONMENTAL

Operating temperature range.	-	0 to 50 deg. C.
Storage temperature range	-	-20 to +70 deg.C.
Humidity	-	0 to 90% non-condensing.
Weight	-	4 oz. (120 gm.)

Appendix C

STORAGE OF INTEGER VARIABLES

Data is stored in integer variables (% type) in 2's complement form. Each integer variable uses 16 bits or 2 bytes of memory. 16 bits of data is equivalent to values from 0 to 65,535 decimal, but the 2's complement convention interprets the most significant bit as a sign bit so the actual range becomes -32,768 to +32,767 (a span of 65,535). Numbers are represented as follows:-

	High byte								Low byte							
	D7	D6	D5	D4	D3	D2	D1	D0	D7	D6	D5	D4	D3	D2	D1	D0
+32,767	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
+10,000	0	0	1	0	0	1	1	1	0	0	0	1	0	0	0	0
+1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
-10,000	1	1	0	1	1	0	0	0	1	1	1	1	0	0	0	0
-32,768	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

^
Sign bit
1 if negative, 0 if positive

Integer variables are the most compact form of storage for the 12 bit data from the A/D converter and 16 bit data of the 8253 interval timer and so to conserve memory and disk space and optimise execution speed, all data exchange with the APM-08 is through integer type variables. This poses a programming problem when handling unsigned numbers in the range 32,768 to 65,535.

If you wish to input or output an unsigned integer greater than 32,767 then it is necessary to work out what its 2's compliment signed equivalent is. As an example, assume we want to load a 16 bit counter with 50,000 decimal.

50,000 (Hex C350) Binary 1100 0011 0101 0000

Since the most significant bit is 1 this would be stored as a negative integer and in fact the correct integer variable value would be 50,000 - 65,536 = -15,536. The programming steps for switching between integer and real variables for representation of unsigned numbers between 0 and 65,535 is therefore:-

From real variable N (0 <= N <= 65,535) to integer variable N%:-

```
xxx10 IF N<=32767 THEN N% = N ELSE N% = N - 65536
```

From integer variable N% to real variable N:-

```
xxx20 IF N% >= 0 THEN N=N% ELSE N = N% + 65536
```

Appendix D

CALIBRATION AND TEST

D.1 CALIBRATION AND TEST

Periodic recalibration of APM-08 is recommended to retain full accuracy. The recalibration interval depends to a large extent on the type of service that the board is subjected to. For an environment with frequent large changes of temperature and/or vibration, a 3 month recalibration interval is recommended. For laboratory or office conditions, 6 months to 1 year is acceptable.

A 4 1/2 digit digital multimeter is required as a minimum equipment to perform a satisfactory calibration. In addition, a voltage calibrator or a stable noise free D.C. voltage source that can be used in conjunction with the digital multimeter is required.

The BASIC listing that appears at the end of this section is useful for calibrating both the A/D and D/A's. and the locations of the calibration adjustments are shown in Fig. D.1

5.1 CALIBRATING THE A/D

The A/D adjustments on APM-08 are 2 trimmer potentiometers that control the A/D - and +.Full Scale. The A/D output should be observed while applying a known calibration voltage to any or all analog input channels. Briefly the adjustment sequence is:-

1. Apply an analog input of -4.9988v and adjust the -F.S. pot so that the output flickers between 0 & 1. This pot is marked R4 on the APM-08 board and the lefthand one.

2. Apply an analog input of +4.9963v and adjust the +F.S. pot so that the output flickers between 4094 and 4095. This pot is marked R5 and is the righthand.

Adjustments are done on "half" bit intervals to obtain a reading flickering about 50/50 between two adjacent values. This is more precise than applying center bit values such as -5.0000v,

+4.9988v etc. and performs a better calibration.

D.2 CALIBRATING THE D/A's

The procedure for adjusting the D/A's is as follows:-

1. Connect the -10v reference to the D/A Ref. Inputs.

2. Output digital zero to the selected D/A and connect the D.V.M to the D/A output and L.L. GND. Adjusting the D/A zero trimpot for a reading of +/-0.0000 v on the D.V.M.

3. Output digital full scale (4095) to the D/A and adjust the D/A full scale adjust trimpot for a reading of +9.9976 v on the D.V.M.

4. You can now output any intermediate code e.g. 2048 for 1/2 scale and check the linearity of the output on the D.V.M. It should be within +/-1.2mV of the theoretical ideal, if not the D/A is probably faulty.

5. Repeat steps 2 thru 4 for the other D/A.

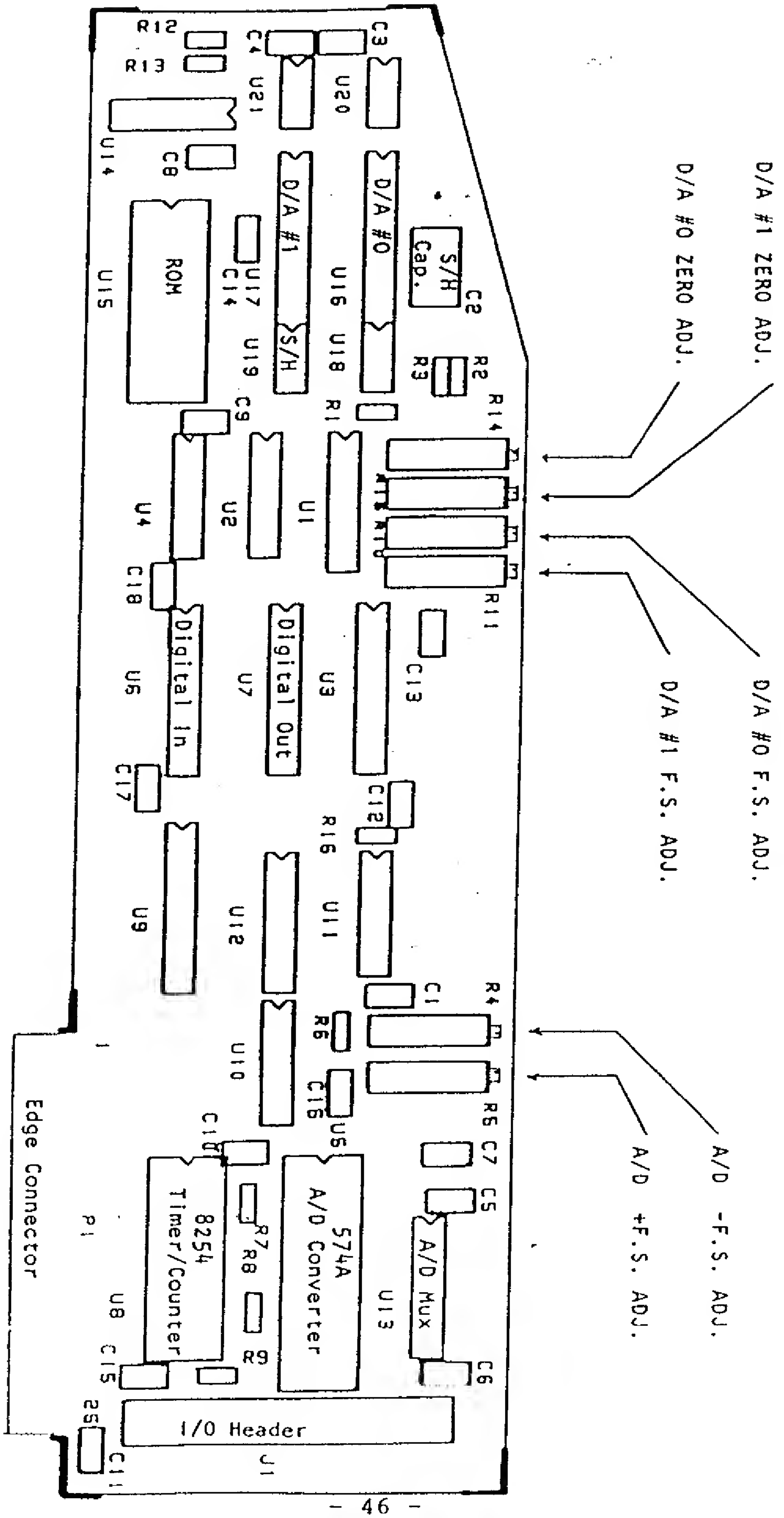


Fig. D.1 APM-08 CALIBRATION ADJUSTMENTS

D.3 BASIC CALIBRATION PROGRAM LISTING

```
*********** APPLESOFT BASIC CALIBRATION PROGRAM LISTING ****************

5      HOME
10     INPUT "ENTER SLOT NUMBER--  ";S%
20     BASADR% =  - 16256 + (S% * 16)
25     HOME
30     PRINT
40     PRINT "CHOOSE FUNCTION TO TEST/CALIBRATE."
45     PRINT
50     PRINT "1 -- A/D CONVERSION"
60     PRINT " "
70     PRINT "2 -- D/A CONVERSION"
80     PRINT
90     PRINT "3 -- COUNTER/TIMER"
100    PRINT
110    PRINT "4 -- DIGITAL INPUT"
120    PRINT
130    PRINT "5 -- DIGITAL OUTPUT"
133    PRINT
134    PRINT "6 -- EXIT ROUTINE"
135    PRINT
140    INPUT "ENTER CHOICE--  ";C
150    IF C = 2 THEN  GOTO 500
152    IF C = 6 THEN  GOTO 1500
160    IF C = 3 THEN  GOTO 800
170    IF C = 4 THEN  GOTO 1100
180    IF C = 5 THEN  GOTO 1400
183    HOME
185    REM   START OF A/D CONVERSION ROUTINES.
186    PRINT
190    INPUT "NUMBER OF CHANNELS TO SCAN--  ";NCH%
193    PRINT
200    INPUT "FIRST CHANNEL TO SCAN --  ";LOWCH%
202    PRINT
205    PRINT "HIT ANY KEY TO CONTINUE"
206    IF ( PEEK ( - 16384)) <  = 127 THEN  GOTO 206
207    HOME : POKE ( - 16368),0
208    PRINT
210    HICH% = LOWCH% + NCH%
223    HOME
224    HTAB 1
227    PRINT "CHANNEL","READING"
228    PRINT
230    FOR CH = LOWCH% TO (HICH% - 1)
240    POKE (BASADR% + 2),CH
250    POKE (BASADR% + 1),0
```

```
260   XL% =  PEEK (BASADR%)
270   XH% =  PEEK (BASADR% + 1)
280   X% = XH% * 16 + XL% / 16
290   PRINT " "
300   PRINT CH,X%
320   NEXT CH
340   PRINT
345   PRINT "HIT ANY KEY TO END SCAN"
347   IF ( PEEK ( - 16384)) >  = 127 THEN  POKE ( - 16368),0: GOTO 25
348   FOR Q = 1 TO 200: NEXT Q
350   GOTO 223
360   GOTO 25
500   HOME
505   VTAB 1: HTAB 1
510   PRINT "0 -- ZERO VOLTS"
515   PRINT
520   PRINT "1 -- HALF SCALE"
525   PRINT
530   PRINT "2 -- FULL SCALE"
535   PRINT
540   PRINT "3 -- EXIT ROUTINE"
550   PRINT
560   INPUT "ENTER CHOICE";C
565   PRINT : PRINT
570   IF C = 0 THEN LOW% = 0:HIGH% = 0: PRINT "ZERO VOLTS OUT"
      : GOTO 650
580   IF C = 1 THEN LOW% = 0:HIGH% = 128: PRINT "HALF SCALE OUT"
      : GOTO 650
590   IF C = 2 THEN LOW% = 255:HIGH% = 255: PRINT "FULL SCALE OUTPUT"
      : GOTO 650
600   IF C = 3 THEN  GOTO 25
610   GOTO 500
650   POKE (BASADR% + 8),LOW%
660   POKE (BASADR% + 10),LOW%
670   POKE (BASADR% + 9),HIGH%
680   POKE (BASADR% + 11),HIGH%
690   GOTO 505
800   HOME
805   VTAB 1: HTAB 1
810   PRINT "COUNTER TIMER TEST SECTION"
820   INPUT "COUNTER TO TEST 0, 1, OR 2--  ";CTR%
825   IF CTR% = 0 THEN CNTL% = 0: GOTO 850
830   IF CTR% = 1 THEN CNTL% = 64: GOTO 850
840   IF CTR% = 2 THEN CNTL% = 128: GOTO 850
845   GOTO 800
850   PRINT "COUNTER BEING TESTED IS COUNTER # ",CTR%
855   REM  SET 2 BYTE XFER
860   CNTL% = CNTL% + 48
870   INPUT "COUNTER MODE";MD%
880   CNTL% = CNTL% + MD% * 2
890   INPUT "DIVISOR FOR COUNTER ";DIV
900   HIGH% =  INT (DIV / 256)
910   LOW% = DIV - HIGH% * 256
920   POKE (BASADR% + 7),CNTL%
```

```
930   POKE (BASADR% + 4 + CTR%),LOW%
940   POKE (BASADR% + 4 + CTR%),HIGH%
941   L =  PEEK (BASADR% + 4 + CTR%)
942   H =  PEEK (BASADR% + 4 + CTR%)
943   PRINT "READ DATA = ";L + H * 256
950   INPUT "CONFIGURE ANOTHER COUNTER?  1-YES, 2-NO ";A
960   IF A = 1 THEN  GOTO 800
970   GOTO 25
1100  HOME
1110  PRINT "INPUT PORT TEST"
1120  INP% =  PEEK (BASADR% + 2)
1130  IF INP% > 128 THEN INP% = INP% - 128
1140  INP% =  INT (INP% / 16)
1150  PRINT INP%
1160  INPUT "ANOTHER READING  1-YES, 2-NO  ";A
1170  IF A = 1 THEN  GOTO 1100
1180  GOTO 25
1400  HOME
1410  PRINT "OUTPUT PORT TEST"
1420  INPUT "ENTER AN OUTPUT # (0 - 15) OR A NEGATIVE # TO EXIT  ";N%
1430  IF N% < 0 THEN  GOTO 25
1440  IF N% > 15 THEN  GOTO 1400
1445  POKE (BASADR% + 2),N% * 16
1450  GOTO 1400
1500  END
```

Appendix E

APM-08 ROM LISTING

AVOCET SYSTEMS 6502 CROSS-ASSEMBLER - VERSION 2.02M

SOURCE FILE NAME: APM08.ASM

```
 0000           ****************APM-08 SOURCE LISTING**************


 0000           *****************************************************
 0000           ****locations $0700 through $0FF are the ***********
 0000           ****256 bytes of relocatable I/O ROM space. ********
 0000           *****************************************************


 C700                   ORG     $C700

 C700           *******SET UP BASE ADRESS LOCATIONS*****************

 C080           BASE0   EQU     $C080
 C081           BASE1   EQU     $C081
 C082           BASE2   EQU     $C082
 C083           BASE3   EQU     $C083
 C084           BASE4   EQU     $C084
 C085           BASE5   EQU     $C085
 C086           BASE6   EQU     $C086
 C087           BASE7   EQU     $C087

 C088           BASE8   EQU     $C088
 C089           BASE9   EQU     $C089
 C08A           BASEA   EQU     $C08A
 C08B           BASEB   EQU     $C08B
 C08C           BASEC   EQU     $C08C
 C08D           BASED   EQU     $C08D
 C08E           BASEE   EQU     $C08E
 C08F           BASEF   EQU     $C08F


 C700           ************SET UP RAM ADRESS LOCATIONS**********
```

```
0478           RAM0    EQU  0478H
04F8           RAM1    EQU  04F8H
0578           RAM2    EQU  0578H
05F8           RAM3    EQU  05F8H
0678           RAM4    EQU  0678H
06F8           RAM5    EQU  06F8H
0778           RAM6    EQU  0778H
07F8           RAM7    EQU  07F8H

C700           ********  SAVE OLD 6502 REGISTER CONTENTS********

C700 48                PHA
C701 8A                TXA
C702 48                PHA
C703 98                TYA
C704 48                PHA
C705 08                PHP

C706           *********DETERMINE THE BOARD'S SLOT ADRESS*******

C706 78                SEI                    ;DISABLE THE INTERUPTS
C707 2058FF            JSR     $FF58
C70A BA                TSX
C70B BD0001            LDA     $0100,X
C70E 8DF807            STA     $07F8
C711 290F              AND     #$0F
C713 A8                TAY          ; Y CONTAINS  0N (N = SLOT #)
C714 58                CLI          ;RE-ENABLE INTERUPTS

C715           ********TURN OFF ALL EXPANSION CARDS*************

C715 2CFFCF            BIT     $CFFF

C718                   ;CLEAR THIS CARDS RAM SCRATCHPAD

C718
C718 A900              LDA     #$00
C71A 997804            STA     RAM0,Y
C71D 99F804            STA     RAM1,Y
C720 997805            STA     RAM2,Y
C723 99F805            STA     RAM3,Y
C726 997806            STA     RAM4,Y
C729 99F806            STA     RAM5,Y
C72C 997807            STA     RAM6,Y
C72F 99F807            STA     RAM7,Y

C732           ****** DETERMINE IF INPUT OR OUTPUT**********
C732
C732 98                TYA
C733 09C0              ORA     #$C0
C735 C537              CMP     $0037
C737 F012              BEQ     ENDOUT ;****JUMP TO OUTPUT ROUTINE
```

```
C739                      ;CSW KSW DECODE AN INPUT

C739 98                   TYA
C73A 09C0                 ORA     #$C0
C73C 8539                 STA     $39              ;SET KSWH TO $CN
C73E A9A0                 LDA     #$A0
C740 8538                 STA     $38              ;SET KSWL TO $A0

C742                      ;RESET 6502 TO PRE ROUTINE CONDITION

C742 28                   PLP
C743 68                   PLA
C744 A8                   TAY
C745 68                   PLA
C746 AA                   TAX
C747 68                   PLA
C748 18                   CLC
C749 9055                 BCC     RELINP           ;GOTO KSW LOCATION

C74B                      ;OUTPUT ROUTINE

C74B 98          ENDOUT:  TYA
C74C 09C0                 ORA     #$C0
C74E 8537                 STA     $37              ;STORE CSWH
C750 A9D0                 LDA     #$D0
C752 8536                 STA     $36              ;STORE CSWL

C754                     ;*****TEST TO SEE IF INPUT VECTOR NEEDS
C754                     ;******TO BE CHANGED TO CNA0

C754 98                   TYA
C755 09C0                 ORA     #$C0
C757 AA                   TAX
C758 E439                 CPX     $39
C75A D004                 BNE     NOINP   ;BRANCH TO NOINP IF IN# N
C75C                                      ;HAS NOT SELECTED THIS SLOT

C75C A9A0                 LDA     #$A0
C75E 8538                 STA     $38              ;STORE KSWL

C760             *********RESET 6502 TO PRE-ROUTINE STATE*********

C760 28          NOINP:   PLP
C761 68                   PLA
C762 A8                   TAY
C763 68                   PLA
C764 AA                   TAX
C765 68                   PLA
C766

C766 18                   CLC
C767 9067                 BCC     RELOUT           ;BRANCH TO $CnD0
```

```
C769      ************************************************************
C769      *******$07A0 IS THE NEW LOCATION OF THE CSWH AND CSWL******
C769      *******POINTERS.  THIS SECTION MUST BE IN THE         ******
C769      *******RELOCATABLE ROM SECTION IN ORDER TO TURN ON THE*****
C769      *******CORRECT EXPANSION ROM.                          *****
C769      ************************************************************
C7A0                      ORG      $C7A0

C7A0 48          RELINP:  PHA
C7A1 8A                   TXA                ;STORE 6502 STATUS
C7A2 48                   PHA
C7A3 98                   TYA
C7A4 48                   PHA
C7A5 08                   PHP
C7A6 4C00C8               JMP      INPUT ;JUMP TO ABSOLUTELY ADRESSED $C800
C7A9                                     ;WHICH IS THE INPUT ROUTINE.


C7A9      ************************************************************
C7A9      ********$07D0 IS THE NEW LOCATION OF THE KSWL AND KSWH*****
C7A9      ******** POINTERS.  THIS SECTION MUST BE IN THE        *****
C7A9      ********RELOCATABLE ROM SECTION IN ORDER TO TURN ON THE****
C7A9      ********CORRECT EXPANSION ROM                          *****
C7A9      ************************************************************
C7D0                      ORG      $C7D0

C7D0 48          RELOUT:  PHA                ;STORE 6502 STATUS
C7D1 8A                   TXA
C7D2 48                   PHA
C7D3 98                   TYA
C7D4 48                   PHA
C7D5 08                   PHP
C7D6 4C87C8               JMP      PRINT ;** GO TO OUTPUT (PRINT) ROUTINE
C7D9                                     ;** IN EXPANSION ROM ADRESS SPACE

C7D9      ;************************************************************
C7D9      ;************************************************************
C7D9      ;********                                              ******
C7D9      ;********   EXPANSION MEMORY ROUTINES  (C800 - CEFF)    ******
C7D9      ;********                                              ******
C7D9      ;************************************************************
C7D9      ;************************************************************


C800                      ORG      $C800

C800 A537        INPUT:   LDA      $37       ;PICK OFF SLOT #
C802 290F                 AND      #$0F
C804 A8                   TAY

C805 B9F807               LDA      RAM7,Y    ;CLEAR RAM7 IF AN ILLEGAL ENTRY
C808 C907                 CMP      #$07      ;HAS BEEN PICKED UP
C80A 9005                 BCC      GOINP
C80C A900                 LDA      #$00
```

```
C80E 99F807              STA     RAM7,Y

C811 B97806     GOINP:   LDA     RAM4,Y   ;SEE IF A MINUS SIGN TO BE
C814                                      ;INPUTED
C814 29F0                AND     #$F0
C816 C9F0                CMP     #$F0
C818 D014                BNE     GOINP1

C81A A9FF                LDA     #$FF     ;USE THE 4 MSB'S OF RAM4 TO SEE
C81C 99F807              STA     RAM7,Y   ;IF A MINUS SIGN NEEDS BE SENT
C81F B97806              LDA     RAM4,Y   ;IF MSB'S = $B THEN NO MINUS SIGN
C822 290F                AND     #$0F
C824 09B0                ORA     #$B0
C826 997806              STA     RAM4,Y
C829 A9AD                LDA     #$AD
C82B 4C71C8              JMP     LOAD

C82E B9F807     GOINP1:  LDA     RAM7,Y   ;BEGIN SENDING ASCII BYTES IN RAM4
C831 C900                CMP     #$00     ;THORUGH RAM0,  KEEPING TRACK OF
C833 D006                BNE     ONE      ;WHICH BYTE TO SEND NEXT BY
C835 B97806              LDA     RAM4,Y   ;INCREMENTING RAM7
C838 4C71C8              JMP     LOAD

C83B B9F807     ONE:     LDA     RAM7,Y
C83E C901                CMP     #$01
C840 D006                BNE     TWO
C842 B9F805              LDA     RAM3,Y
C845 4C71C8              JMP     LOAD

C848 B9F807     TWO:     LDA     RAM7,Y
C84B C902                CMP     #$02
C84D D006                BNE     THREE
C84F B97805              LDA     RAM2,Y
C852 4C71C8              JMP     LOAD

C855 B9F807     THREE:   LDA     RAM7,Y
C858 C903                CMP     #$03
C85A D006                BNE     FOUR
C85C B9F804              LDA     RAM1,Y
C85F 4C71C8              JMP     LOAD

C862 B9F807     FOUR:    LDA     RAM7,Y
C865 C904                CMP     #$04
C867 D006                BNE     CR
C869 B97804              LDA     RAM0,Y
C86C 4C71C8              JMP     LOAD


C86F A98D       CR:      LDA     #$8D

C871 BA         LOAD:    TSX     ;MOVE STACK POINTER AND STUFF
C872 E8                  INX     ;BYTE TO BE INPUTED INTO WHAT WILL BE
C873 E8                  INX     ;THE ACCUMULATOR AT THE END OR THE
C874 E8                  INX     ;6502 RESTORE FUNCTION EXIT.
```

```
C875 E8                 INX
C876 9A                 TXS
C877 48                 PHA
C878 68                 PLA
C879 CA                 DEX
C87A CA                 DEX
C87B CA                 DEX
C87C CA                 DEX
C87D 9A                 TXS


C87E 18                 CLC
C87F B9F807             LDA     RAM7,Y  ;INCREMENT RAM7 TO KEEP TRACK OF
C882 6901               ADC     #$01    ;WHICH BYTE TO INPUT NEXT
C884 99F807             STA     RAM7,Y

C887  ;************************************************************
C887  ;******  PRINT ROUTINE                                 *******
C887  ;************************************************************

C887                    ;** PICK OFF PRINTED ASCII BYTE**

C887   *****AT THE START OF EACH ROM ROUTINE ALL CURRENT PROCESSOR****
C887   *****STATUS ITEMS ARE PUSHED ONTO THE STACK... THUS TO HAVE****
C887   *****THE ACCUMULATOR AT THE TIME OF THE PRINT AVAILABLE YOU****
C887   *****MUST DIG IT OUT OF THE BOTTOM OF THE STACK            ****

C887 BA         PRINT:  TSX     ;LOAD STACK POINTER
C888 E8                 INX
C889 E8                 INX
C88A E8                 INX
C88B 9A                 TXS     ;POINT TO DATA BYTE
C88C 68                 PLA     ;POP DATA
C88D 48                 PHA
C88E CA                 DEX     ;RESET STACK POINTER
C88F CA                 DEX
C890 CA                 DEX
C891 9A                 TXS
C892
C892  ************BEGIN ACTUAL DECODE OPERATION

C892  ******NOTE THAT THE APPLE PRINTS CERTAIN ITEMS WHEN IN****
C892  ******THE INPUT FUNCTION.  CERTAIN PRECAUTIONS MUST BE****
C892  ******TAKEN TO ASSURE THESE PRINTED BYTES ARE NOT ********
C892  ******INTERPRETTED AS PRINT COMMANDS *********************
C892

C892 AA                 TAX     ;PUT ASCII BYTE IN X
C893 E0BF               CPX     #$BF ;IGNORE THE ? SENT BY THE INPUT
C895 F024               BEQ     EXIT2
```

```
C897 E089                 CPX     #$89     ;IGNORE TABS
C899 F020                 BEQ     EXIT2
C89B E0A0                 CPX     #$A0     ;IGNORE SPACES
C89D F01C                 BEQ     EXIT2

C89F A537                 LDA     $37      ;STORE SLOT # IN Y
C8A1 290F                 AND     #$0F
C8A3 A8                   TAY

C8A4 B9F807               LDA     RAM7,Y
C8A7 C906                 CMP     #$06 ;IF THIS IS THE LAST TERM INPUTED
C8A9 F013                 BEQ     CLR7     ;CLR RAM7 AND EXIT


C8AB C900                 CMP     #$00     ;IF THIS IS NOT A NEW COMMAND
C8AD D015                 BNE     FNSCAN   ;BRANCH TO SCAN ROUTINE
C8AF

C8AF 8A                   TXA
C8B0 C98D                 CMP     #$8D     ;IGNORE FIRST CR AFTER INPUT
C8B2 F007                 BEQ     EXIT2    ;COMMAND
C8B4 C9AD                 CMP     #$AD     ;IGNORE MINUS SIGN AS A COMMAND
C8B6 F003                 BEQ     EXIT2

C8B8 99F807               STA     RAM7,Y

C8BB 4CA5C9     EXIT2:    JMP     EXIT     ;SIMPLE BRANCH TO ALLOW EXIT

C8BE 201BCB     CLR7:     JSR     RESET    ;CLEAR ALL RAM LOCATIONS IF
C8C1 4CA5C9               JMP     EXIT     ; INPUT IS OVER


C8C4 B9F807     FNSCAN:   LDA     RAM7,Y
C8C7 C9C3                 CMP     #$C3     ;IF NOT PERFORMING AN A TO D
C8C9 D003                 BNE     NOTC     ;CONVERSION GO TO NOTC
C8CB 4C09C9               JMP     ATOD

C8CE C9C4       NOTC:     CMP     #$C4     ;IF NOT A D TO A CONVERSION
C8D0 D003                 BNE     NOTD     ;GO TO NOTD
C8D2 4C70CA               JMP     DTOA

C8D5 C9CF       NOTD:     CMP     #$CF
C8D7 D003                 BNE     NOTO     ;DIGITAL OUTPUTS?
C8D9 4C33CB               JMP     DIGOUT

C8DC C9C9       NOTO:     CMP     #$C9     ; DIGITAL INPUTS?
C8DE D003                 BNE     NOTI
C8E0 4C77CB               JMP     DIGIN

C8E3 C9D2       NOTI:     CMP     #$D2     ;COUNTER READ?
C8E5 D003                 BNE     NOTR
C8E7 4CACCB               JMP     RDCTR

C8EA C9D7       NOTR:     CMP     #$D7     ;COUNTER LOAD?
```

```
C8EC D003                 BNE     NOTW
C8EE 4C16CC               JMP     LDCTR

C8F1 C9CC       NOTW:     CMP     #$CC      ;COUNTER LATCH?
C8F3 D003                 BNE     NOTL
C8F5 4CA0CC               JMP     CTRLT

C8F8 C9D3       NOTL:     CMP     #$D3      ;SET COUNTER MODE?
C8FA D003                 BNE     NOTS
C8FC 4CD3CC               JMP     CTRMD

C8FF C9D1       NOTS:     CMP     #$D1      ;EXIT TO DOS ROUTINE?
C901 D003                 BNE     NOTQ
C903 4C19CD               JMP     QUIT

C906 4CA5C9     NOTQ:     JMP     EXIT ;NOT A VALID COMMAND AND SO IGNORED

C909 E08D       ATOD:     CPX     #$8D      ;IF "CR"  GO PERFORM CONVERSION
C90B F00E                 BEQ     CONV

C90D B97807               LDA     RAM6,Y    ;IF THIS IS THE SECOND CHANNEL
C910 C900                 CMP     #$00      ;NUMBER SELECTED THEN IGNORE IT.
C912 D004                 BNE     EXIT1


C914 8A                   TXA               ;STORE CHANNEL # IN
C915 997807               STA     RAM6,Y    ;RAM6
C918 4CA5C9     EXIT1:    JMP     EXIT

C91B            **********START CONVERSION ROUTINE****************

C91B B97807     CONV:     LDA     RAM6,Y

C91E 2907                 AND     #$07 ;PICK OFF CHANNEL NUMBER TO CONVERT
C920 AA                   TAX               ;STORE CHANNEL IN X

C921 B9F806               LDA     RAM5,Y    ;PICK OFF DIGITAL OUTPUT PORT
C924 29F8                 AND     #$F8
C926 99F806               STA     RAM5,Y
C929 8A                   TXA
C92A 19F806               ORA     RAM5,Y
C92D 99F806               STA     RAM5,Y    ;MERGE CHANNEL # ON END OF
C930                                        ;CONTROL WORD
C930 48                   PHA
C931 98                   TYA               ;MOVE ON IN Y TO NO
C932 2A                   ROL     A
C933 2A                   ROL     A
C934 2A                   ROL     A
C935 2A                   ROL     A
C936 29F0                 AND     #$F0      ;CLEAR LSB'S
C938 A8                   TAY
C939 68                   PLA
C93A 9982C0               STA     BASE2,Y ;WRITE NEW CONTROL REGISTER
```

```
C93D A900                LDA     #$00
C93F EA        STALL:    NOP          ;ALLOW S+H SOME TIME TO ACQUIRE
C940 6901                ADC     #$01
C942 C904                CMP     #$04
C944 D0F9                BNE     STALL

C946 9981C0              STA     BASE1,Y ;START A/D CONVERSION
C949 98                  TYA
C94A 48                  PHA
C94B
C94B 4A                  LSR     A
C94C 4A                  LSR     A
C94D 4A                  LSR     A
C94E 4A                  LSR     A
C94F 290F                AND     #$0F
C951 A8                  TAY
C952 A900                LDA     #$00
C954 997804              STA     RAM0,Y
C957 99F804              STA     RAM1,Y
C95A 997805              STA     RAM2,Y
C95D 99F805              STA     RAM3,Y
C960 997806              STA     RAM4,Y
C963 997807              STA     RAM6,Y
C966 99F807              STA     RAM7,Y
C969 68                  PLA
C96A AA                  TAX              ;STORE N0 IN REGISTER X

C96B  **************SET UP SCRATCH PAD RAM FOR TOBCD SUBROUTINE**
C96B  ********THE TOBCD ROUTINE TAKES BINARY DATA FROM**********
C96B  ********RAM6, AND RAM7 AND CONVERTS IT INTO ASCII*********
C96B  ********DATA IN RAM0 THROUGH RAM4  (5 DIGITS)*************
C96B  ********WHERE RAM0 IS THE LSD AND RAM4 IS THE MSD*********
C96B  ********IF THE NUMBER IS > THAN 32767 THEN THE MSB'S******
C96B  ********OF RAM4 ARE SET TO ONES TO TELL THE INPUT COMMAND*
C96B  ********TO SEND A MINUS SIGN BEFORE THE 5 DIGIT NUMBER****
C96B  ********THE ROUTINE THEN SUBTRACTS 32768 FROM THE INPUT***
C96B  ********SO WE CAN MAINTAIN COMPATABILITY WITH THE BINARY**
C96B  ********INTEGER FORMAT  ***********************************

C96B BD80C0              LDA     BASE0,X    ;PICK OF LSB'S
C96E 4A                  LSR     A
C96F 4A                  LSR     A
C970 4A                  LSR     A
C971 4A                  LSR     A
C972 997807              STA     RAM6,Y

C975 BD81C0              LDA     BASE1,X     ;PICK OFF MSB'S
C978 4A                  LSR     A
C979 4A                  LSR     A
C97A 4A                  LSR     A
C97B 4A                  LSR     A
C97C 290F                AND     #$0F
C97E 99F807              STA     RAM7,Y
```

```
C981 BD81C0              LDA     BASE1,X
C984 2A                  ROL     A
C985 2A                  ROL     A
C986 2A                  ROL     A
C987 2A                  ROL     A
C988 29F0                AND     #$F0
C98A 197807              ORA     RAM6,Y
C98D 997807              STA     RAM6,Y

C990 98                  TYA
C991 AA                  TAX
C992 20ADC9              JSR     TOBCD
C995 A537                LDA     $37       ;RESET Y TO THE BOARD #
C997 290F                AND     #$0F
C999 A8                  TAY
C99A A900                LDA     #$00
C99C 99F807              STA     RAM7,Y
C99F 997807              STA     RAM6,Y

C9A2 4CA5C9              JMP     EXIT
C9A5


C9A5 18         EXIT:    CLC
C9A6 28                  PLP               ;RESTORE 6502
C9A7 68                  PLA
C9A8 A8                  TAY
C9A9 68                  PLA
C9AA AA                  TAX
C9AB 68                  PLA
C9AC 60                  RTS


C9AD  ***********BINARY TO ASCII CONVERSION ROUTINE*******
C9AD  *****THIS ROUTINE TAKES BCD NUMBERS IN RAM6 (LSD) THROUGH
C9AD  *****RAM7 (MSD) AND CONVERTS THEM INTO ASCII CHARACTERS
C9AD  *****IN THE SAME MEMORY LOCATIONS-- REGISTER X MUST CONTAIN
C9AD  *****ON BEFORE CALLING THIS ROUTINE.

C9AD B9F807     TOBCD:   LDA     RAM7,Y
C9B0 2980                AND     #$80
C9B2 C900                CMP     #$00
C9B4 F00B                BEQ     TENTHO    ;GOTO TENTHO IF THE # IS POSITIVE

C9B6  *********NEGATIVE NUMBER***********************************
C9B6 B9F807              LDA     RAM7,Y
C9B9 297F                AND     #$7F
C9BB 99F807              STA     RAM7,Y
C9BE A9FF                LDA     #$FF      ;PUT 01 IN Y IF NEGATIVE
C9C0 A8                  TAY

C9C1
C9C1 38         TENTHO:  SEC                         ;COUNT HOW MANY TIMES
```

```
C9C2 BD7807             LDA     RAM6,X              ;10,000 CAN BE SUBTRACTED
C9C5 E910               SBC     #$10                ;FROM THE NUMBER TO FIND
C9C7 48                 PHA                         ;THE TEN THOUSANDS
C9C8 BDF807             LDA     RAM7,X
C9CB E927               SBC     #$27
C9CD 48                 PHA
C9CE
C9CE 900E               BCC     THOUSA    ;IF NEGATIVE THEN STORE TENTHOUSANDS
C9D0 FE7806             INC     RAM4,X    ;AND GOTO SUBTRACTION ROUTINE
C9D3 68                 PLA               ;FOR THOUSANDS
C9D4 9DF807             STA     RAM7,X
C9D7 68                 PLA
C9D8 9D7807             STA     RAM6,X
C9DB 4CC1C9             JMP     TENTHO    ;IF NOT NEGATIVE GOT BACK AND DO
C9DE                                      ;ANOTHER SUBTRACTION

C9DE 68         THOUSA: PLA               ;COUNT HOW MANY THOUSANDS CAN BE
C9DF 68                 PLA               ;SUBTRACTED FORM THE RESULT
C9E0 38         THOUS:  SEC
C9E1 BD7807             LDA     RAM6,X
C9E4 E9E8               SBC     #$E8
C9E6 48                 PHA
C9E7 BDF807             LDA     RAM7,X
C9EA E903               SBC     #$03
C9EC 48                 PHA
C9ED 900E               BCC     HUNSA
C9EF FEF805             INC     RAM3,X
C9F2 68                 PLA
C9F3 9DF807             STA     RAM7,X
C9F6 68                 PLA
C9F7 9D7807             STA     RAM6,X
C9FA 4CE0C9             JMP     THOUS

C9FD 68         HUNSA:  PLA               ;COUNT HOW MANY HUNDREDS
C9FE 68                 PLA
C9FF 38         HUNS:   SEC
CA00 BD7807             LDA     RAM6,X
CA03 E964               SBC     #$64
CA05 48                 PHA
CA06 BDF807             LDA     RAM7,X
CA09 E900               SBC     #$00
CA0B 48                 PHA
CA0C 900E               BCC     TENSA
CA0E FE7805             INC     RAM2,X
CA11 68                 PLA
CA12 9DF807             STA     RAM7,X
CA15 68                 PLA
CA16 9D7807             STA     RAM6,X
CA19 4CFFC9             JMP     HUNS

CA1C 68         TENSA:  PLA               ;COUNT THE # OF TENS
CA1D 68                 PLA
CA1E 38         TENS:   SEC
CA1F BD7807             LDA     RAM6,X
```

```
CA22 E90A               SBC     #$0A
CA24 48                 PHA
CA25 900A               BCC     ONESA
CA27 FEF804             INC     RAM1,X
CA2A 68                 PLA
CA2B 9D7807             STA     RAM6,X
CA2E 4C1ECA             JMP     TENS

CA31 68       ONESA:    PLA                 ;HOW MANY ONES
CA32 BD7807   ONES:     LDA     RAM6,X
CA35 9D7804             STA     RAM0,X

CA38 98                 TYA
CA39 C9FF               CMP     #$FF
CA3B D008               BNE     GONES
CA3D BD7806             LDA     RAM4,X
CA40 09F0               ORA     #$F0
CA42 9D7806             STA     RAM4,X

CA45 8A       GONES:    TXA                 ;OR A $B ONTO THE MSB'S OF THE ASCII
CA46 A8                 TAY                 ;TO GET TRUE APPLE ASCII
CA47 B97806             LDA     RAM4,Y
CA4A 09B0               ORA     #$B0
CA4C 997806             STA     RAM4,Y
CA4F B9F805             LDA     RAM3,Y
CA52 09B0               ORA     #$B0
CA54 99F805             STA     RAM3,Y
CA57 B97805             LDA     RAM2,Y
CA5A 09B0               ORA     #$B0
CA5C 997805             STA     RAM2,Y
CA5F B9F804             LDA     RAM1,Y
CA62 09B0               ORA     #$B0
CA64 99F804             STA     RAM1,Y
CA67 B97804             LDA     RAM0,Y
CA6A 09B0               ORA     #$B0
CA6C 997804             STA     RAM0,Y

CA6F 60                 RTS

CA70 E08D     DTOA:     CPX     #$8D
CA72 F02E               BEQ     LDDAC    ;GO TO LLDAC IF "CR"

CA74 B97807             LDA     RAM6,Y   ; IF 1ST NUMBER STORE IN RAM4
CA77 C900               CMP     #$00     ;ELSE GOTO DAC1
CA79 D00E               BNE     DAC1     ;THIS NUMBER WILL BE THE DAC
CA7B 8A                 TXA              ;CHANNEL NUMBER
CA7C 290F               AND     #$0F
CA7E 997806             STA     RAM4,Y
CA81 A901               LDA     #$01
CA83 997807             STA     RAM6,Y

CA86 4CA5C9             JMP     EXIT

CA89   ***********LOAD NEW BYTE IN MEMORY AND INCREMENT LOLD BYTES
```

```
CA89                                      ;MOVES EVERYTHING UP ONE
CA89                                      ;BYTE.  THIS MEANS ONLY THE
CA89 B97805    DAC1:    LDA       RAM2,Y  ;LAST 5 BYTES INPUTED
CA8C 99F805             STA       RAM3,Y  ;WILL BE READ
CA8F B9F804             LDA       RAM1,Y
CA92 997805             STA       RAM2,Y
CA95 B97804             LDA       RAM0,Y
CA98 99F804             STA       RAM1,Y
CA9B 8A                 TXA
CA9C 997804             STA       RAM0,Y
CA9F 4CA5C9             JMP       EXIT

CAA2 B97806    LDDAC:   LDA       RAM4,Y  ;BEGIN LOADING THE DAC ROUTINES
CAA5 AA                 TAX               ;PUT DAC NUMBER IN X
CAA6 A900               LDA       #$00
CAA8 997806             STA       RAM4,Y

CAAB 2089CD             JSR       BCDBIN  ; GO TO THE ASCII/BINARY CONVERT

CAAE B97807             LDA       RAM6,Y   ;SHIFT RAM6 AND 7 TO PROPER
CAB1                                       ;CONFIGURATION
CAB1 0A                 ASL       A
CAB2 997807             STA       RAM6,Y
CAB5 B9F807             LDA       RAM7,Y
CAB8 2A                 ROL       A
CAB9 99F807             STA       RAM7,Y
CABC B97807             LDA       RAM6,Y  ;SECOND TWO BYTE SHIFT
CABF 0A                 ASL       A
CAC0 997807             STA       RAM6,Y
CAC3 B9F807             LDA       RAM7,Y
CAC6 2A                 ROL       A
CAC7 99F807             STA       RAM7,Y

CACA B97807             LDA       RAM6,Y  ;THIRD TWO BYTE SHIFT
CACD 0A                 ASL       A
CACE 997807             STA       RAM6,Y
CAD1 B9F807             LDA       RAM7,Y
CAD4 2A                 ROL       A
CAD5 99F807             STA       RAM7,Y

CAD8 B97807             LDA       RAM6,Y  ;FOURTH SHIFT
CADB 0A                 ASL       A
CADC 997807             STA       RAM6,Y
CADF B9F807             LDA       RAM7,Y
CAE2 2A                 ROL       A
CAE3 99F807             STA       RAM7,Y

CAE6           ********** SET UP FOR ACTAUL DAC LOAD***********

CAE6 8A                 TXA               ;STORE  CHANNEL #
CAE7 997804             STA       RAM0,Y
CAEA 98                 TYA
CAEB 0A                 ASL       A
```

```
CAEC 0A                 ASL     A
CAED 0A                 ASL     A
CAEE 0A                 ASL     A        ;MOVE  NO INTO X
CAEF AA                 TAX

CAF0 B97804             LDA     RAM0,Y
CAF3 C900               CMP     #$00     ;BRANCH IF WRITTING TO DAC1
CAF5 D012               BNE     LDDAC1
CAF7 B97807             LDA     RAM6,Y
CAFA 9D88C0             STA     BASE8,X ;WRITE LSB'S OF DAC0
CAFD B9F807             LDA     RAM7,Y
CB00 9D89C0             STA     BASE9,X ;WRITE MSB'S OF DAC0
CB03 201BCB             JSR     RESET

CB06 4CA5C9             JMP     EXIT

CB09 B97807     LDDAC1: LDA     RAM6,Y
CB0C 9D8AC0             STA     BASEA,X         ;LOAD DAC1 LSB'S
CB0F B9F807             LDA     RAM7,Y
CB12 9D8BC0             STA     BASEB,X         ;LOAD DAC1 MSB'S
CB15 201BCB             JSR     RESET


CB18 4CA5C9             JMP     EXIT


CB1B A900       RESET:  LDA     #$00
CB1D 997804             STA     RAM0,Y
CB20 99F804             STA     RAM1,Y
CB23 997805             STA     RAM2,Y
CB26 99F805             STA     RAM3,Y
CB29 997806             STA     RAM4,Y
CB2C 997807             STA     RAM6,Y
CB2F 99F807             STA     RAM7,Y

CB32 60                 RTS
CB33            **********************************************
CB33            *********START OF DIGITAL OUT ROUTINE*********
CB33            **********************************************
CB33 8A         DIGOUT: TXA
CB34 C98D               CMP     #$8D     ;IF 'CR' THEN WRITE TOPORT
CB36 F00D               BEQ     LDDOUT
CB38
CB38 B97804             LDA     RAM0,Y
CB3B 99F804             STA     RAM1,Y
CB3E 8A                 TXA
CB3F 997804             STA     RAM0,Y
CB42 4CA5C9             JMP     EXIT

CB45 A900       LDDOUT: LDA     #$00
CB47 997805             STA     RAM2,Y
CB4A 99F805             STA     RAM3,Y
CB4D 997806             STA     RAM4,Y
CB50
```

```
CB50 2089CD              JSR    BCDBIN  ;GOSUB ASCII/BINARY CONVERTER

CB53 98                  TYA            ;SET UP FOR WRITE TO BOARD
CB54 0A                  ASL    A
CB55 0A                  ASL    A
CB56 0A                  ASL    A
CB57 0A                  ASL    A
CB58 AA                  TAX

CB59  **********  SET UP CONTROL REGISTER FOR WRITE TO APM08*****

CB59 B97807              LDA    RAM6,Y
CB5C 0A                  ASL    A
CB5D 0A                  ASL    A
CB5E 0A                  ASL    A
CB5F 0A                  ASL    A
CB60 99F805              STA    RAM3,Y
CB63 B9F806              LDA    RAM5,Y
CB66 290F                AND    #$0F    ;CLEAR MSB'S OF OLD CONTROL REGISTER
CB68 19F805              ORA    RAM3,Y  ;OR IN NEW OUTPUT WORD
CB6B 99F806              STA    RAM5,Y
CB6E 9D82C0              STA    BASE2,X ;WRITE TO CONTROL REGISTER
CB71 201BCB              JSR    RESET
CB74 4CA5C9              JMP    EXIT


CB77            ************************************************
CB77            ********START OF DIGITAL INPUT ROUTINE***********
CB77            ************************************************

CB77 8A         DIGIN:   TXA
CB78 C98D                CMP    #$8D    ;IF CR THEN READ PORT
CB7A F003                BEQ    INPRD   ;IF NOT EXIT
CB7C 4CA5C9              JMP    EXIT

CB7F 98         INPRD:   TYA            ;SET UP FOR READ
CB80 0A                  ASL    A
CB81 0A                  ASL    A
CB82 0A                  ASL    A
CB83 0A                  ASL    A
CB84 AA                  TAX
CB85 BD82C0              LDA    BASE2,X
CB88 4A                  LSR    A
CB89 4A                  LSR    A
CB8A 4A                  LSR    A
CB8B 4A                  LSR    A
CB8C 2907                AND    #$07
CB8E 09B0                ORA    #$B0
CB90 997804              STA    RAM0,Y
CB93 A9B0                LDA    #$B0
CB95 99F804              STA    RAM1,Y
CB98 997805              STA    RAM2,Y
CB9B 99F805              STA    RAM3,Y
CB9E 997806              STA    RAM4,Y
```

```
CBA1 A900               LDA     #$00
CBA3 997807             STA     RAM6,Y
CBA6 99F807             STA     RAM7,Y
CBA9 4CA5C9             JMP     EXIT


CBAC             ***************************************************
CBAC             *********START COUNTER READ ROUTINE**************
CBAC             ***************************************************

CBAC E08D        RDCTR: CPX     #$8D
CBAE F00E               BEQ     READ

CBB0 B97807             LDA     RAM6,Y  ;IF SECOND DIGIT OF COUNTER TO
CBB3 C900               CMP     #$00    ;READ IGNORE
CBB5 D004               BNE     EXIT3
CBB7 8A                 TXA
CBB8 997807             STA     RAM6,Y  ;STORE COUNTER TO READ IN RAM6


CBBB 4CA5C9      EXIT3: JMP EXIT

CBBE B97807      READ:  LDA     RAM6,Y  ;STORE COUNTER TO READ IN RAM0
CBC1 2903               AND     #$03    ;PICK OFF COUNTER #
CBC3 997804             STA     RAM0,Y

CBC6 98                 TYA
CBC7 0A                 ASL     A
CBC8 0A                 ASL     A
CBC9 0A                 ASL     A
CBCA 0A                 ASL     A
CBCB AA                 TAX

CBCC B97804             LDA     RAM0,Y  ;GET COUNTER #
CBCF C902               CMP     #$02
CBD1 F022               BEQ     RDCTR2
CBD3 C901               CMP     #$01
CBD5 F00F               BEQ     RDCTR1

CBD7             ***********READ COUNTER 0***************
CBD7 BD84C0             LDA     BASE4,X ;GET LSB'S OF COUNTER 0
CBDA 997807             STA     RAM6,Y
CBDD BD84C0             LDA     BASE4,X ;GET MSB'S OF COUNTER 0
CBE0 99F807             STA     RAM7,Y

CBE3 4C01CC             JMP     RDCONV

CBE6             *********READ COUNTER 1 ****************
CBE6 BD85C0      RDCTR1: LDA    BASE5,X
CBE9 997807             STA     RAM6,Y
CBEC BD85C0             LDA     BASE5,X
CBEF 99F807             STA     RAM7,Y

CBF2 4C01CC             JMP     RDCONV
```

```
CBF5           *********READ COUNTER 2******************
CBF5 BD86C0    RDCTR2: LDA     BASE6,X
CBF8 997807            STA     RAM6,Y
CBFB BD86C0            LDA     BASE6,X
CBFE 99F807            STA     RAM7,Y


CC01 98        RDCONV: TYA
CC02 AA                TAX
CC03 20ADC9            JSR     TOBCD     ;GO TO BINARY/ASCII CONVERTER
CC06 A537              LDA     $37       ;ROUTINE
CC08 290F              AND     #$0F
CC0A A8                TAY
CC0B A900              LDA     #$00
CC0D 99F807            STA     RAM7,Y
CC10 997807            STA     RAM6,Y
CC13 4CA5C9            JMP     EXIT


CC16           *************************************************
CC16           ********START OF COUNTER LOAD ROUTINE************
CC16           *************************************************


CC16 E08D      LDCTR:  CPX     #$8D
CC18 F02D              BEQ     LDCNTR    ;GO TO LDCNTR IF "CR"

CC1A B97807            LDA     RAM6,Y    ; IF COUNTER NUMBER STORE IN RAM4
CC1D C900              CMP     #$00      ;ELSE GOTO LD1
CC1F D007              BNE     LD1       ;THIS NUMBER WILL BE THE CNTR
CC21 8A                TXA               ;CHANNEL NUMBER
CC22 997807            STA     RAM6,Y

CC25 4CA5C9            JMP     EXIT


CC28    ***********************************************************
CC28    ***********LOAD NEW BYTE IN MEMORY AND INCREMENT LOLD BYTES
CC28    ********AT THIS POINT THE ASCII VERSION OF THE COUNTER #**
CC28    ********IS STORED IN RAM6  *******************************
CC28    **********************************************************

CC28                                     ;MOVES EVERYTHING UP ONE
CC28                                     ;BYTE.  THIS MEANS ONLY THE
CC28 B9F805    LD1:    LDA     RAM3,Y    ;LAST 5 DIGITS INPUTTED WILL
CC2B 997806            STA     RAM4,Y    ;BE RECOGNIZED AS INPUT
CC2E B97805            LDA     RAM2,Y
CC31 99F805            STA     RAM3,Y
CC34 B9F804            LDA     RAM1,Y
CC37 997805            STA     RAM2,Y
CC3A B97804            LDA     RAM0,Y
CC3D 99F804            STA     RAM1,Y
CC40 8A                TXA
```

```
CC41 997804             STA     RAM0,Y
CC44 4CA5C9             JMP     EXIT

CC47 B97807     LDCNTR: LDA     RAM6,Y
CC4A AA                 TAX               ;STORE CNTR # IN X

CC4B 2089CD             JSR     BCDBIN    ;GO TO ASCII/BINARY CONVERTER
CC4E                                      ;ROUTINE
CC4E A537               LDA     $37
CC50 290F               AND     #$0F
CC52 A8                 TAY
CC53 8A                 TXA
CC54 997804             STA     RAM0,Y    ;STORE COUNTER # IN RAM0
CC57
CC57 98                 TYA               ;SET UP INDEX FOR WRITE TO APM08
CC58 0A                 ASL     A
CC59 0A                 ASL     A
CC5A 0A                 ASL     A
CC5B 0A                 ASL     A
CC5C AA                 TAX

CC5D B97804             LDA     RAM0,Y
CC60 2903               AND     #$03
CC62 C902               CMP     #$02
CC64 F028               BEQ     CTR2LD    ;IF LOADING TO CTR 2 GOTO CTR2LD
CC66 C901               CMP     #$01
CC68 F012               BEQ     CTR1LD    ;IF LOADING TO CTR 1 GOTO CTR1LD

CC6A            **********LOAD CTR # 0*********************
CC6A B97807             LDA     RAM6,Y    ;GET LSB'S
CC6D 9D84C0             STA     BASE4,X   ;WRITE LSB'S TO CTR 0
CC70 B9F807             LDA     RAM7,Y    ;GET MSB'S
CC73 9D84C0             STA     BASE4,X   ;WRITE MSB'S TO CTR 0
CC76 201BCB             JSR     RESET
CC79 4CA5C9             JMP     EXIT


CC7C            *********LOAD CTR # 1***********************
CC7C B97807     CTR1LD: LDA     RAM6,Y    ;GET LSB'S
CC7F 9D85C0             STA     BASE5,X   ;WRITE LSB'S TO CTR 0
CC82 B9F807             LDA     RAM7,Y    ;GET MSB'S
CC85 9D85C0             STA     BASE5,X   ;WRITE MSB'S TO CTR 0
CC88 201BCB             JSR     RESET
CC8B 4CA5C9             JMP     EXIT


CC8E            ********LOAD CTR # 2************************
CC8E B97807     CTR2LD: LDA     RAM6,Y    ;GET LSB'S
CC91 9D86C0             STA     BASE6,X   ;WRITE LSB'S TO CTR 0
CC94 B9F807             LDA     RAM7,Y    ;GET MSB'S
CC97 9D86C0             STA     BASE6,X   ;WRITE MSB'S TO CTR 0
CC9A 201BCB             JSR     RESET
CC9D 4CA5C9             JMP     EXIT
```

```
CCA0             *************************************************
CCA0             ******START OF COUNTER LATCH ROUTINE*************
CCA0             *************************************************

CCA0  E08D       CTRLT:  CPX      #$8D     ;GO LATCH ON CR
CCA2  F00E               BEQ      LATCH
CCA4  B97807             LDA      RAM6,Y
CCA7  C900               CMP      #$00
CCA9  D004               BNE      EXIT4
CCAB  8A                 TXA
CCAC  997807             STA      RAM6,Y   ;STORE COUNTER NUMBER IN RAM6
CCAF  4CA5C9     EXIT4:  JMP EXIT

CCB2  B97807     LATCH:  LDA      RAM6,Y   ;PICK OFF COUNTER NUMBER AND
CCB5  0A                 ASL      A        ;STORE IN RAM0
CCB6  0A                 ASL      A
CCB7  0A                 ASL      A
CCB8  0A                 ASL      A
CCB9  0A                 ASL      A
CCBA  0A                 ASL      A
CCBB  29C0               AND      #$C0
CCBD  997807             STA      RAM6,Y
CCC0  98                 TYA
CCC1  0A                 ASL      A        ;SET UP INDEX FOR WRITE TO APM08
CCC2  0A                 ASL      A
CCC3  0A                 ASL      A
CCC4  0A                 ASL      A
CCC5  AA                 TAX
CCC6  B97807             LDA      RAM6,Y
CCC9  EA                 NOP
CCCA  9D87C0             STA      BASE7,X
CCCD  201BCB             JSR      RESET

CCD0  4CA5C9             JMP      EXIT

CCD3  E08D       CTRMD:  CPX      #$8D
CCD5  F01C               BEQ      SETMD    ;GO TO SETMD IF "CR"

CCD7  B97807             LDA      RAM6,Y   ;IF 1ST NUMBER STORE IN RAM4
CCDA  C900               CMP      #$00     ;ELSE GOTO GETMD
CCDC  D00E               BNE      GETMD    ;THIS NUMBER WILL BE THE COUNTER
CCDE  8A                 TXA               ;NUMBER
CCDF  290F               AND      #$0F
CCE1  997806             STA      RAM4,Y
CCE4  A901               LDA      #$01
CCE6  997807             STA      RAM6,Y
CCE9  4CA5C9             JMP      EXIT

CCEC  8A         GETMD:  TXA
CCED  997804             STA      RAM0,Y   ;STORE COUNTER MODE IN RAM0
CCF0  4CA5C9             JMP      EXIT

CCF3  ************************************************************
```

```
CCF3   ***************AT THIS POINT WE HAVE THE COUNTER NUMBER*****
CCF3   **************STORED IN RAM4 AND THE MODE STORED IN RAM0***
CCF3   ***********************************************************


CCF3 98          SETMD:  TYA                ;SET UP FOR WRITE TO BOARD
CCF4 0A                  ASL     A
CCF5 0A                  ASL     A
CCF6 0A                  ASL     A
CCF7 0A                  ASL     A
CCF8 AA                  TAX
CCF9 B97806              LDA     RAM4,Y     ;SHIFT COUNTER NUMBER INTO
CCFC 0A                  ASL     A          ;MOST SIGNIFICANT TWO BITS
CCFD 0A                  ASL     A
CCFE 0A                  ASL     A
CCFF 0A                  ASL     A
CD00 0A                  ASL     A
CD01 0A                  ASL     A
CD02 997806              STA     RAM4,Y
CD05 B97804              LDA     RAM0,Y     ;SHIFT MODE NUMBER ONE BIT TO
CD08 2907                AND     #$07       ;THE LEFT
CD0A 0A                  ASL     A
CD0B 197806              ORA     RAM4,Y     ;MERGE CNTR # AND MODE #
                                            ;IN CNTRL REG
CD0E 0930                ORA     #$30       ;SET COUNTER FO 2 BYTE TRANSFER
CD10 9D87C0              STA     BASE7,X    ;WRITE COUNTER CONTROL WORD

CD13 201BCB              JSR     RESET      ;CLEAR RAM

CD16 4CA5C9              JMP EXIT

CD19   ************************************************************
CD19   ****** QUIT ROUTINE TO SET UP DOS **************************
CD19   ************************************************************


CD19 8A          QUIT:   TXA                ;RESET EXPANSION CARD SPACE SO
CD1A C98D                CMP     #$8D       ;DOS CAN BE ACTIVATED
CD1C F003                BEQ     SETBIT
CD1E 4CA5C9              JMP     EXIT
CD21 201BCB      SETBIT: JSR     RESET      ;REINITIALIZE THE SETTINGS FOR
CD24 A99E                LDA     #$9E       ;DOS INPUTS AND OUTPUTS
CD26 8537                STA     $37
CD28 8539                STA     $39
CD2A A9BD                LDA     #$BD
CD2C 8536                STA     $36
CD2E A981                LDA     #$81
CD30 8538                STA     $38
CD32 4CA5C9              JMP     EXIT

CD35 18          MULT:   CLC           ;SIMPLE TIMES TEN MULTIPLICATION
CD36 B9F805              LDA     RAM3,Y  ;ROUTINE
CD39 0A                  ASL     A
CD3A 99F805              STA RAM3,Y
```

```
CD3D B97806             LDA RAM4,Y
CD40 2A                 ROL      A
CD41 997806             STA      RAM4,Y
CD44 48                 PHA
CD45 B9F805             LDA      RAM3,Y
CD48 48                 PHA
CD49 B9F805             LDA      RAM3,Y
CD4C 0A                 ASL      A
CD4D 99F805             STA      RAM3,Y
CD50 B97806             LDA      RAM4,Y
CD53 2A                 ROL      A
CD54 997806             STA      RAM4,Y
CD57 B9F805             LDA      RAM3,Y
CD5A 0A                 ASL      A
CD5B 99F805             STA      RAM3,Y
CD5E B97806             LDA      RAM4,Y
CD61 2A                 ROL      A
CD62 997806             STA      RAM4,Y
CD65 18                 CLC
CD66 68                 PLA
CD67 79F805             ADC      RAM3,Y
CD6A 99F805             STA      RAM3,Y
CD6D 68                 PLA
CD6E 797806             ADC      RAM4,Y
CD71 997806             STA      RAM4,Y

CD74 60                 RTS


CD75 18        ADD:     CLC      ;SIMPLE TWO BYTE ADDITION ROUTINE
CD76 B9F805             LDA      RAM3,Y
CD79 797807             ADC      RAM6,Y
CD7C 997807             STA      RAM6,Y
CD7F B97806             LDA      RAM4,Y
CD82 79F807             ADC      RAM7,Y
CD85 99F807             STA      RAM7,Y

CD88 60                 RTS

CD89  *****START OF BCD TO BINARY CONVERTER ROUTINE***********

CD89  ***********************************************************
CD89  *****THIS ROUTINE TAKES ASCII DATA FROM RAM4 (MSD) THROUGH*
CD89  *****RAM0 (LSD) AND CONVERTS IT INTO BCD DATA IN RAM7     *
CD89  *****AND RAM7                                             *
CD89  ***********************************************************

CD89 B97804    BCDBIN:  LDA      RAM0,Y  ;TAKE DATA FROM RAM4 THROUGH RAM0
CD8C 290F               AND      #$0F    ;AND STORE IT IN A COMPACTED FORM IN
CD8E 997804             STA      RAM0,Y  ;RAM0 THROUGH RAM3  (THUS FREEING
CD91 B9F804             LDA      RAM1,Y  ;UP RAM3 AND RAM4 TO BE USED IN
CD94 0A                 ASL      A       ;THE COMPUTATIONS)
CD95 0A                 ASL      A
CD96 0A                 ASL      A
```

```
CD97 0A             ASL    A
CD98 197804         ORA    RAM0,Y
CD9B 997804         STA    RAM0,Y
CD9E B97805         LDA    RAM2,Y
CDA1 290F           AND    #$0F
CDA3 99F804         STA    RAM1,Y
CDA6 B9F805         LDA    RAM3,Y
CDA9 0A             ASL    A
CDAA 0A             ASL    A
CDAB 0A             ASL    A
CDAC 0A             ASL    A
CDAD 19F804         ORA    RAM1,Y
CDB0 99F804         STA    RAM1,Y
CDB3 B97806         LDA    RAM4,Y
CDB6 290F           AND    #$0F
CDB8 997805         STA    RAM2,Y


CDBB A900           LDA    #$00
CDBD 99F807         STA    RAM7,Y
CDC0 997806         STA    RAM4,Y
CDC3 B97804         LDA    RAM0,Y
CDC6 290F           AND    #$0F
CDC8 997807         STA    RAM6,Y   ;STORE ONES IN RAM6
CDCB B97804         LDA    RAM0,Y
CDCE 4A             LSR    A
CDCF 4A             LSR    A
CDD0 4A             LSR    A
CDD1 4A             LSR    A
CDD2 99F805         STA    RAM3,Y   ;STORE TENS IN RAM3
CDD5 2035CD         JSR    MULT     ;MULTIPLY BY TEN AND ADD
                                    ;TO THE ONES
CDD8 2075CD         JSR    ADD
CDDB B9F804         LDA    RAM1,Y
CDDE 290F           AND    #$0F
CDE0 99F805         STA    RAM3,Y
CDE3 A900           LDA    #$00
CDE5 997806         STA    RAM4,Y   ;MULTIPLY THE 100'S BYTE BY 100
CDE8 2035CD         JSR    MULT
CDEB 2035CD         JSR    MULT
CDEE 2075CD         JSR    ADD      ;ADD THE RESULT TO THE PREVIOUS DATA
CDF1 B9F804         LDA    RAM1,Y
CDF4 4A             LSR    A
CDF5 4A             LSR    A
CDF6 4A             LSR    A
CDF7 4A             LSR    A
CDF8 99F805         STA    RAM3,Y
CDFB A900           LDA    #$00
CDFD 997806         STA    RAM4,Y
CE00 2035CD         JSR    MULT     ;MULTIPLYT THE THOUSANDS BYTE BY
CE03 2035CD         JSR    MULT     ;1000
CE06 2035CD         JSR    MULT
CE09 2075CD         JSR    ADD      ;ADD THIS TO OUR SUM
CE0C B97805         LDA    RAM2,Y
```

```
 CE0F 99F805              STA      RAM3,Y
 CE12 A900                LDA      #$00
 CE14 997806              STA      RAM4,Y
 CE17 2035CD              JSR      MULT       ;MULT THE 10,000'S
 CE1A 2035CD              JSR      MULT
 CE1D 2035CD              JSR      MULT
 CE20 2035CD              JSR      MULT
 CE23 2075CD              JSR      ADD        ;ADD IT TO OUR SUM
 CE26                                         ;FOR THE COMPLETE BINARY RESULT
 CE26 60                  RTS


 0000                     END

***** NO ERRORS DETECTED *****
```

Index